jQuery UI API 中文文档 1.10

wizardforcel

Published with GitBook



目錄

jQuery UI API 中文文档 1.10

来源：jQuery UI API中文文档

拍砖纠错请微博上@愚人码头。

索引

Effects

jQuery UI 在jQuery 内置的特效上添加了一些功能。jQuery UI 支持颜色动画和 Class 转换，同时也提供了一些额外的 Easings。另外，提供了一套完整的定制特效，供显示和隐藏元素时或者只是添加一些视觉显示时使用。

Also in: Effects Core | Method Overrides

.addClass()

当动画样式改变时，为匹配的元素集合内的每个元素添加指定的 Class。

Blind Effect

百叶窗特效（Blind Effect）通过将元素包裹在一个容器内，采用"拉百叶窗"效果来隐藏或显示元素。

Bounce Effect

反弹特效（Bounce Effect）反弹一个元素。当与隐藏或显示一起使用时，最后一次或第一次反弹会呈现淡入/淡出效果。

Clip Effect

剪辑特效（Clip Effect）通过垂直或水平方向夹剪元素来隐藏或显示一个元素。

Color Animation

使用 .animate() 实现颜色动画效果。

Drop Effect

降落特效（Drop Effect）通过单个方向滑动的淡入淡出来隐藏或显示一个元素。

Easings

Easing 函数指定动画在不同点上的行进速度。

Also in: Effects Core | Methods

.effect()

对一个元素应用动画特效。

Explode Effect

爆炸特效（Explode Effect）通过把元素裂成碎片来隐藏或显示一个元素。

Fade Effect

淡入淡出特效（Fade Effect）通过淡入淡出元素来隐藏或显示一个元素。

Fold Effect

折叠特效（Fold Effect）通过折叠元素来隐藏或显示一个元素。

Also in: Effects Core | Method Overrides | Methods

.hide()

使用自定义效果来隐藏匹配的元素。

Highlight Effect

突出特效（Highlight Effect）通过首先改变背景颜色来隐藏或显示一个元素。

Puff Effect

通过在缩放元素的同时隐藏元素来创建膨胀特效（Puff Effect）。

Pulsate Effect

跳动特效（Pulsate Effect）通过跳动来隐藏或显示一个元素。

Also in: Effects Core | Method Overrides

.removeClass()

当动画样式改变时，为匹配的元素集合内的每个元素移除指定的 Class。

Scale Effect

按照某个百分比缩放元素。

Shake Effect

垂直或水平方向多次震动元素。

Also in: Effects Core | Method Overrides | Methods

.show()

使用自定义效果来显示匹配的元素。

Size Effect

调整元素尺寸到指定宽度和高度。

Slide Effect

把元素滑动出视区。

Blind EffectAlso in: Effects Core

.switchClass()

当动画样式改变时，为匹配的元素集合内的每个元素添加和移除指定的 Class。

Also in: Effects Core | Method Overrides | Methods

.toggle()

使用自定义效果来显示或隐藏匹配的元素。

Also in: Effects Core | Method Overrides

.toggleClass()

当动画样式改变时，根据 Class 是否存在以及 switch 参数的值，为匹配的元素集合内的每个元素添加或移除一个或多个 Class。

Transfer Effect

把一个元素的轮廓转移到另一个元素。

.addClass()

Categories: Effects | Effects Core | Method Overrides

.addClass(className [, duration] [, easing] [, complete])Returns: jQuery

描述: 为每个匹配的元素添加指定的样式类名，而且所有改变的样式以动画的形式展示

- **.addClass(className [, duration] [, easing] [, complete])**
  - **className**Type: String为每个匹配元素的class属性增加一个或多个样式名（空格隔开）。
  - **duration** (default: `400`)Type: Number or String一个字符串或者数字决定动画将运行多久。（愚人码头注：默认值: "normal"， 三种预定速度的字符串("slow", "normal", 或 "fast")或表示动画时长的毫秒数值(如：1000) ）
  - **easing** (default: `swing`)Type: String一个字符串，表示过渡使用哪种缓动 函数。
  - **complete**Type: Function()在动画完成时执行的函数。

- **.addClass(className [, options])**
  - **className**Type: String为每个匹配元素的class属性增加一个或多个样式名（空格隔开）。
  - **options**Type: Object一组包含动画选项的值的集合。 支持的选项:
    - **duration** (default: `400`)Type: Number or String一个字符串或者数字决定动画将运行多久。（译者注：默认值: "normal"， 三种预定速度的字符串("slow", "normal", 或 "fast")或表示动画时长的毫秒数值(如：1000) ）
    - **easing** (default: `swing`)Type: String一个字符串，表示过渡使用哪种缓动 函数。
    - **complete**Type: Function()在动画完成时执行的函数。
    - **children** (default: `false`)Type: Boolean动画是否应用到所有匹配的元素的后代元素。应慎使用此功能。 因为要确定哪些后代元素要应用动画可能会非常好性能， 而且 后代元素的数量是线性增长的。
    - **queue** (default: `true`)Type: Boolean or String一个布尔值，指示是否将动画放置在效果队列中。如果为false时，将立即开始动画。 从**jQuery1.7**开始， 队列选项也可以接受一个字符串，在这种情况下， 在动画被添加到由该字符串表示的队列中。

类似原生CSS transitions（过渡）， jQuery UI的样式动画提供了一个从一个状态到另一个状态平滑过渡， 同时让你 保持那些CSS样式改版的所有细节，和你的javaScript分离。所有样式动画的方法，包括 `.addClass()` 支持自定义的持续时间（durations）和缓冲函数（easing），以及在动画完成时提供一个回调。

并非所有的样式都可以设置动画。 例如，没有办法让背景图像应用动画。不能进行动画的样式，将在动画结束时被改变。

这个插件扩展了jQuery中的 `.addClass()` 方法。 如果没有加载 jQuery UI， 调用 `.addClass()` 方法可能不会失败，因为方法仍然存在。 然而，预期的行为（愚人码头注：指平滑过渡动画效果）将不会发生。

Example:

匹配的元素上添加**"big-blue"**样式。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>addClass demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  div {
    width: 100px;
    height: 100px;
    background-color: #ccc;
  }
  .big-blue {
    width: 200px;
    height: 200px;
    background-color: #00f;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<div></div>

<script>
$( "div" ).click(function() {
  $( this ).addClass( "big-blue", 1000, "easeOutBounce" );
});
</script>

</body>
</html>
```

Blind Effect

Categories: Effects

Description: 百叶窗效果隐藏或者显示一个包装在一个容器中的元素时候具有“拉百叶窗”的效果

- **blind**
  - **direction** (default: `"up"`)Type: String

    direction参数是表示百叶窗效果向哪个方向隐藏元素，或者从哪个方向开始显示元素。

    可选值: `up` , `down` , `left` , `right` , `vertical` , `horizontal` .

容器元素的应用了 `overflow: hidden` 属性，所以高度的变化将影响其可见性。

Example:

使用百叶窗效果切换隐藏显示div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>blind demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<p>Click anywhere to toggle the box.</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "blind" );
});
</script>

</body>
</html>
```

Bounce Effect

Categories: Effects

Description: 反弹特效上下反弹一个元素。当反弹特效和隐藏或显示配合使用时，最后一个或第一个反弹也将呈现淡入或淡出效果。

- **bounce**
  - **distance** (default: `20`)Type: Number反弹的最大距离，单位为像素。
  - **times** (default: `5`)Type: Integer元素将会反弹的次数。当和隐藏或显示搭配使用时，淡入淡出时会有额外的“半”反弹效果。

Example:

使用反弹特效来切换显示**div**。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>bounce demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<p>Click anywhere to toggle the box.</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "bounce", { times: 3 }, "slow" );
});
</script>

</body>
</html>
```

Clip Effect

Categories: Effects

Description: 剪辑效果通过垂直或水平剪辑一个元素来显示或隐藏它，并且它是从元素的两端同时进行的。

- **clip**
  - **direction** (default: `"up"`)Type: String

    剪辑效果将隐藏或显示其中元素的平面。

    `vertical` 值将剪辑顶部和底部边缘, while `horizontal` 值将剪辑左部和右部边缘

Example:

使用剪辑特效切换显示**div**。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>clip demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<p>Click anywhere to toggle the box.</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "clip" );
});
</script>

</body>
</html>
```

Color Animation

jQuery UI特效内核使用 `rgb()` ， `rgba()` ，16进制值，甚至颜色名称比如 `"aqua"` 给颜色增加了动画新特性。只需要包含 jQuery UI特效内核文件和 `.animate()` 就可以获得对各种颜色的支持。

支持以下属性:

- `backgroundColor`
- `borderBottomColor`
- `borderLeftColor`
- `borderRightColor`
- `borderTopColor`
- `color`
- `columnRuleColor`
- `outlineColor`
- `textDecorationColor`
- `textEmphasisColor`

对颜色动画的技术支持来自 jQuery Color plugin。颜色插件提供了多个函数来处理各种颜色。查看完整文档，请访问jQuery Color documentation。

Class Animations（与动画关联的CSS类）

当为单独的颜色属性直接呈现动画效果时，将样式包含在css类中是一个很好的处理方式。jQuery UI提供了一系列方法，这些方法可以根据新增的CSS类来呈现动画，还可以删除任何CSS类。 这些方法包含 `.addClass()` ， `.removeClass()` ， `.toggleClass()` 和 `.switchClass()` 。 这些方法将自动决定哪些属性需要被改变并且对其应用适当的动画。

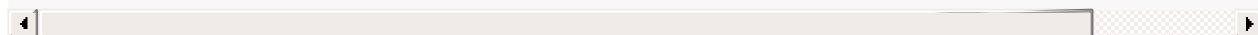
Example

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Color Animation Demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  #elem {
    color: #006;
    background-color: #aaa;
    font-size: 25px;
    padding: 1em;
    text-align: center;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<div id="elem">color animations</div>
<button id="toggle">animate colors</button>

<script>
$( "#toggle" ).click(function() {
  $( "#elem" ).animate({
    color: "green",
    backgroundColor: "rgb( 20, 20, 20 )"
  });
});
</script>

</body>
</html>
```



Drop Effect

Categories: Effects

Description: 拉拽特效隐藏或显示一个元素，并使用通过淡入或淡出效果使它向一个方向滑动。

- **drop**
    - **direction** (default: `"left"`)Type: String

      direction参数表示将向哪个方向隐藏这个元素，或从哪个方向开始显示这个元素。

      可选值：`up`，`down`，`left`，`right`。

Example:

使用拉拽特效切换显示**div**。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>drop demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<p>Click anywhere to toggle the box.</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "drop" );
});
</script>

</body>
</html>
```

Easings

动画缓冲函数可以确定在动画过程中不同时刻的动画速度。 jQuery内核自带有2种动画渐变: `linear`，整个动画在一个恒定的速度中进行（匀速），和 `swing` (jQuery内核的默认渐变效果)，此效果开始和结束的速度要比中间过程的速度要慢。 jQuery UI还提供了其它一些额外的渐变函数，这些函数会在 `swing` 效果中搜索变量来自定义其它效果，比如bouncing（反弹）。

一些缓冲函数在动画过程中会产生负数值。呈现动画效果的属性不同， 某些属性的实际值可能会在为0时被锁定。例如，你可以将 `left` 值渐变为一个负数，但是你不能将 `height` 或者 `opacity` 的值缓冲为一个负数。

理解动画缓冲如何影响动画最好的方式就是查看对应的函数图像。下面列出了jQuery UI中所有动画的函数图像。

.effect()

Categories: Effects | Effects Core | Methods

.effect(effect [, options] [, duration] [, complete])Returns: jQuery

Description: 在一个元素上应用一个动画特效。

- **.effect(effect [, options] [, duration] [, complete])**
  - **effect**Type: String一个字符串，指示哪个特效被使用。
  - **options**Type: Object特效选项和缓冲函数。
  - **duration** (default: `400`)Type: Number or String一个字符串或者数字决定动画将运行多久。（愚人码头注：三种预定速度的字符串("slow", "normal", 或 "fast")或表示动画时长的毫秒数值(如：1000)）
  - **complete**Type: Function()在动画完成时执行的函数。

- **.effect(options)**
  - **options**Type: Object全部动画选项。 只有一个必须设置的属性 `effect` 。
    - **effect**Type: String一个字符串，指示哪个特效被使用。
    - **easing** (default: `"swing"`)Type: String一个字符串，表示过渡使用哪种 缓动 函数。
    - **duration** (default: `400`)Type: Number or String一个字符串或者数字决定动画将运行多久。（愚人码头注：三种预定速度的字符串("slow", "normal", 或 "fast")或表示动画时长的毫秒数值(如：1000)）
    - **complete**Type: Function()在动画完成时执行的函数。
    - **queue** (default: `true`)Type: Boolean or String一个布尔值，指示是否将动画放置在效果队列中。如果为false时，将立即开始动画。 从**jQuery1.7**开始，队列选项也可以接受一个字符串，在这种情况下， 在动画被添加到由该字符串表示的队列中。

`.effect()` 方法将对一个元素应用一个特效。 许多特效同时也具有显示和隐藏效果， 你可以搭配使用 `.show()` ，`.hide()` ，和 `.toggle()` 方法来实现。

Example:

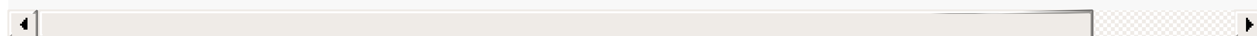
一个**div**应用反弹特效。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>effect demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  div {
    width: 100px;
    height: 100px;
    background: #ccc;
    border: 1px solid #000;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<p>Click anywhere to apply the effect.</p>
<div></div>

<script>
$( document ).click(function() {
  $( "div" ).effect( "bounce", "slow" );
});
</script>

</body>
</html>
```



Explode Effect

分类: 特效

描述: 爆炸特效通过将元素分拆成若干片来隐藏或显示一个元素。

- **explode**
  - **pieces** (默认值: `9`)类型: 整型要分裂的数量， 应该是一个完全平方数,任何其它值都会被四舍五入转换为最接近的完全平方数。

例子:

使用分裂特效切换显示**div**。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>explode demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<p>点击任何区域来切换显示div。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "explode" );
});
</script>

</body>
</html>
```

Fade Effect

Categories: Effects

Description: 淡入淡出的隐藏或显示一个元素。.

- **fade**

Example:

使用淡入淡出特效切换显示**div**。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>fade demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<p>Click anywhere to toggle the box.</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "fade" );
});
</script>

</body>
</html>
```

Fold Effect

Categories: Effects

Description: 通过折叠形式来隐藏或显示一个元素。

- **fold**
    - **size** (default: `15`)Type: Number or String折叠元素的尺寸。
    - **horizFirst** (default: `false`)Type: Boolean当先从水平方向开始隐藏时，记住，当显示时，方向是和隐藏时相反的。（愚人码头注：即如果隐藏时为从右至左，那么显示时为从左至右。）

Example:

使用折叠特效切换显示**div**。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>fold demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<p>Click anywhere to toggle the box.</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "fold" );
});
</script>

</body>
</html>
```

.hide()

Categories: Effects | Effects Core | Method Overrides | Methods

.hide(effect [, options] [, duration] [, complete])Returns: jQuery

Description: 使用自定义的效果隐藏匹配的元素。

- **.hide(effect [, options] [, duration] [, complete])**
  - **effect**Type: String一个字符串，指示哪个特效被使用。
  - **options**Type: Object特效选项和缓冲函数。
  - **duration** (default: `400`)Type: Number or String一个字符串或者数字决定动画将运行多久。（愚人码头注：三种预定速度的字符串("slow", "normal", 或 "fast")或表示动画时长的毫秒数值(如：1000)）
  - **complete**Type: Function()在动画完成时执行的函数。

- **.hide(options)**
  - **options**Type: Object全部动画选项。 只有一个必须设置的属性 `effect` 。
    - **effect**Type: String一个字符串，指示哪个特效被使用。
    - **easing** (default: `"swing"`)Type: String一个字符串，表示过渡使用哪种 缓动 函数。
    - **duration** (default: `400`)Type: Number or String一个字符串或者数字决定动画将运行多久。（愚人码头注：三种预定速度的字符串("slow", "normal", 或 "fast")或表示动画时长的毫秒数值(如：1000)）
    - **complete**Type: Function()在动画完成时执行的函数。
    - **queue** (default: `true`)Type: Boolean or String一个布尔值，指示是否将动画放置在效果队列中。如果为false时，将立即开始动画。 从**jQuery1.7**开始，队列选项也可以接受一个字符串，在这种情况下， 在动画被添加到由该字符串表示的队列中。

这个插件扩展了jQuery中的 `.hide()` 方法。 如果没有加载 jQuery UI，调用 `.hide()` 方法可能不会失败，因为方法仍然存在。 然而，预期的行为（愚人码头注：指平滑过渡动画效果）将不会发生。

Example:

使用拖动特效切换显示div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>hide demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  div {
    width: 100px;
    height: 100px;
    background: #ccc;
    border: 1px solid #000;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<button>hide the div</button>
<div></div>

<script>
$( "button" ).click(function() {
  $( "div" ).hide( "drop", { direction: "down" }, "slow" );
});
</script>

</body>
</html>
```

Highlight Effect

Categories: Effects

Description: 高亮特效通过先动画呈现元素的背景颜色来隐藏或显示一个元素。

- **highlight**
    - **color** (default: `"#ffff99"`)Type: String动画执行时的背景颜色。

Example:

使用高亮特效切换显示**div**。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>highlight demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<p>Click anywhere to toggle the box.</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "highlight" );
});
</script>

</body>
</html>
```

Puff Effect

Categories: Effects

Description: 创建拉伸的效果，缩放元素的同时隐藏或显示元素。

- **puff**
  - **percent** (default: `150`)Type: Number元素放大的百分比。

Example:

使用拉伸特效来切换显示**div**。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>puff demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<p>Click anywhere to toggle the box.</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "puff" );
});
</script>

</body>
</html>
```

Pulsate Effect

Categories: Effects

Description: 闪烁效果通过脉冲闪烁隐藏或显示元素。

- **pulsate**
  - **times** (default: `5`)Type: Integer元素闪烁的次数，在隐藏或显示时会额外增加一个半闪烁效果。

Example:

使用闪烁特效来切换显示**div**。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>pulsate demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<p>Click anywhere to toggle the box.</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "pulsate" );
});
</script>

</body>
</html>
```

.removeClass()

Categories: Effects | Effects Core | Method Overrides

.removeClass(className [, duration] [, easing] [, complete])Returns: jQuery

Description: 为每个匹配的元素移除指定的样式类名，而且所有改变的样式以动画的形式展示

- **.removeClass(className [, duration] [, easing] [, complete])**
  - **className**Type: String为每个匹配元素的class属性移除一个或多个样式名（空格隔开）。
  - **duration** (default: `400`)Type: Number or String一个字符串或者数字决定动画将运行多久。（愚人码头注：默认值: "normal"， 三种预定速度的字符串("slow", "normal", 或 "fast")或表示动画时长的毫秒数值(如：1000) ）
  - **easing** (default: `swing`)Type: String一个字符串，表示过渡使用哪种缓动 函数。
  - **complete**Type: Function()在动画完成时执行的函数。
- **.removeClass(className [, options])**
  - **className**Type: String为每个匹配元素的class属性移除一个或多个样式名（空格隔开）。
  - **options**Type: Object一组包含动画选项的值的集合。 支持的选项:
    - **duration** (default: `400`)Type: Number or String一个字符串或者数字决定动画将运行多久。（译者注：默认值: "normal"， 三种预定速度的字符串("slow", "normal", 或 "fast")或表示动画时长的毫秒数值(如：1000) ）
    - **easing** (default: `swing`)Type: String一个字符串，表示过渡使用哪种缓动 函数。
    - **complete**Type: Function()在动画完成时执行的函数。
    - **children** (default: `false`)Type: Boolean动画是否应用到所有匹配的元素的后代元素。应慎使用此功能。 因为要确定哪些后代元素要应用动画可能会非常好性能，而且 后代元素的数量是线性增长的。
    - **queue** (default: `true`)Type: Boolean or String一个布尔值，指示是否将动画放置在效果队列中。如果为false时，将立即开始动画。 从**jQuery1.7**开始，队列选项也可以接受一个字符串，在这种情况下， 在动画被添加到由该字符串表示的队列中。

类似原生CSS transitions（过渡）， jQuery UI的样式动画提供了一个从一个状态到另一个状态平滑过渡， 同时让你 保持那些CSS样式改版的所有细节，和你的javaScript分离。所有样式动画的方法，包括 `.removeClass()` 支持自定义的持续时间（durations）和缓冲函数（easing），以及在动画完成时提供一个回调。

并非所有的样式都可以设置动画。 例如，没有办法让背景图像应用动画。不能进行动画的样式，将在动画结束时被改变。

这个插件扩展了jQuery中的 `.removeClass()` 方法。 如果没有加载 jQuery UI，调用 `.removeClass()` 方法可能不会失败，因为方法仍然存在。 然而，预期的行为（愚人码头注：指平滑过渡动画效果）将不会发生。

Example:

匹配的元素上移除"big-blue"样式。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>removeClass demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  div {
    width: 100px;
    height: 100px;
    background-color: #ccc;
  }
  .big-blue {
    width: 200px;
    height: 200px;
    background-color: #00f;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<div class="big-blue"></div>

<script>
$( "div" ).click(function() {
  $( this ).removeClass( "big-blue", 1000, "easeInBack" );
});
</script>

</body>
</html>
```

Scale Effect

Categories: Effects

Description: 按百分比缩小或放大一个元素。

- **scale**
  - **direction** (default: `"both"`)Type: String效果的方向。 可选值: `"both"` , `"vertical"` or `"horizontal"` .
  - **origin** (default: `[ "middle", "center" ]`)Type: Array消失点。
  - **percent**Type: Number缩放百分比。
  - **scale** (default: `"both"`)Type: String确定元素的哪个区域将被缩放: `"both"` , `"box"` , `"content"` 。 Box值缩放元素的border和padding; content值缩放元素内部的所有内容。

Examples:

Example: 使用缩放特效切换显示div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>scale demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<p>Click anywhere to toggle the box.</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "scale" );
});
</script>

</body>
</html>
```

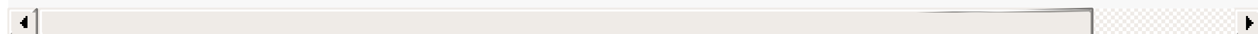
Example: 使用缩放特效，但只朝一个方向切换显示div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>scale demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<p>Click anywhere to toggle the box.</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle({ effect: "scale", direction: "horizontal" });
});
</script>

</body>
</html>
```



Shake Effect

Categories: Effects

Description: 在垂直或水平方向上多次抖动一个元素。

- **shake**
  - **direction** (default: `"left"`)Type: String一个"left"或"right"值将使元素在水平方向抖动， "up"或"down"值将使元素在垂直方向抖动。 该值指定元素这个效果应沿轴线移动的第一步的方向。
  - **distance** (default: `20`)Type: Number抖动距离。
  - **times** (default: `3`)Type: Integer抖动次数。

Example:

抖动一个div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>shake demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<p>Click anywhere to shake the box.</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).effect( "shake" );
});
</script>

</body>
</html>
```

.show()

Categories: Effects | Effects Core | Method Overrides | Methods

.show(effect [, options] [, duration] [, complete])Returns: jQuery

Description: 使用自定义特效显示匹配的元素。

- **.show(effect [, options] [, duration] [, complete])**
  - **effect**Type: String一个字符串，指示哪个特效被使用。
  - **options**Type: Object特效选项和缓冲函数。
  - **duration** (default: `400`)Type: Number or String一个字符串或者数字决定动画将运行多久。（愚人码头注：三种预定速度的字符串("slow", "normal", 或 "fast")或表示动画时长的毫秒数值(如：1000) ）
  - **complete**Type: Function()在动画完成时执行的函数。

- **.show(options)**
  - **options**Type: Object全部动画选项。 只有一个必须设置的属性 `effect` 。
    - **effect**Type: String一个字符串，指示哪个特效被使用。
    - **easing** (default: `"swing"`)Type: String一个字符串，表示过渡使用哪种 缓动 函数。
    - **duration** (default: `400`)Type: Number or String一个字符串或者数字决定动画将运行多久。（愚人码头注：三种预定速度的字符串("slow", "normal", 或 "fast")或表示动画时长的毫秒数值(如：1000) ）
    - **complete**Type: Function()在动画完成时执行的函数。
    - **queue** (default: `true`)Type: Boolean or String一个布尔值，指示是否将动画放置在效果队列中。如果为false时，将立即开始动画。 从**jQuery1.7**开始， 队列选项也可以接受一个字符串，在这种情况下， 在动画被添加到由该字符串表示的队列中。

这个插件扩展了jQuery中的 `.show()` 方法。 如果没有加载 jQuery UI，调用 `.show()` 方法可能不会失败，因为方法仍然存在。 然而，预期的行为（愚人码头注：指平滑过渡动画效果）将不会发生。

Example:

使用折叠特效显示**div**。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>show demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  div {
    display: none;
    width: 100px;
    height: 100px;
    background: #ccc;
    border: 1px solid #000;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<button>show the div</button>
<div></div>

<script>
$( "button" ).click(function() {
  $( "div" ).show( "fold", 1000 );
});
</script>

</body>
</html>
```

Size Effect

Categories: Effects

Description: 将元素的尺寸设置到一个指定的宽度和高度。

- **size**
  - **to**Type: Object改变后高和宽。
  - **origin** (default: `[ "top", "left" ]`)Type: Array特效起始的方向。
  - **scale** (default: `"both"`)Type: String确定元素的哪个区域将被缩放: `"both"`, `"box"`, `"content"`。Box值缩放元素的border和padding; content值缩放元素内部的所有内容。

Example:

使用尺寸特效来改变**div**的尺寸。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>size demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<p>Click anywhere to resize the box.</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).effect( "size", {
    to: { width: 200, height: 60 }
  }, 1000 );
});
</script>

</body>
</html>
```

Slide Effect

Categories: Effects

Description: 将元素滑动到可视区域之外。

- **slide**
  - **direction** (default: `"both"`)Type: String特效的执行方向。可选值: `"left"` , `"right"` , `"up"` , `"down"` 。
  - **distance** (default: `element's outerWidth` （元素的外边缘宽）)Type: Number特效执行的距离。 默认值为元素的高或宽,取决于 `direction` 参数的值来决定（愚人码头注：水平方向为宽，垂直方向为高）。 可以被设置为任何小于元素宽/高的整数。

Example:

使用滑动特效切换显示**div**。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>slide demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<p>Click anywhere to toggle the box.</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "slide" );
});
</script>

</body>
</html>
```

.switchClass()

Categories: Effects | Effects Core

.switchClass(removeClassName, addClassName [, duration] [, easing] [, complete])Returns: jQuery

Description: 为每一组匹配的元素添加或移除指定的样式类，而且所有改变的样式以动画的形式展示

- **.switchClass(removeClassName, addClassName [, duration] [, easing] [, complete])**
  - **removeClassName**Type: String将要被从匹配的元素移除的class属性的一个或多个名称(以空格分开)。
  - **addClassName**Type: String将要被添加到匹配的元素的class属性的一个或多个名称(以空格分开)。
  - **duration** (default: `400`)Type: Number or String一个字符串或者数字决定动画将运行多久。（译者注：默认值: "normal"， 三种预定速度的字符串("slow", "normal", 或 "fast")或表示动画时长的毫秒数值(如：1000) ）
  - **easing** (default: `swing`)Type: String一个字符串，表示过渡使用哪种缓动 函数。
  - **complete**Type: Function()在动画完成时执行的函数。
- **.switchClass(removeClassName, addClassName [, options])**
  - **removeClassName**Type: String将要被从匹配的元素移除的class属性的一个或多个名称(以空格分开)。
  - **addClassName**Type: String将要被添加到匹配的元素的class属性的一个或多个名称(以空格分开)。
  - **options**Type: Object一组包含动画选项的值的集合。所有属性是可选择的。
    - **duration** (default: `400`)Type: Number or String一个字符串或者数字决定动画将运行多久。（译者注：默认值: "normal"， 三种预定速度的字符串("slow", "normal", 或 "fast")或表示动画时长的毫秒数值(如：1000) ）
    - **easing** (default: `swing`)Type: String一个字符串，表示过渡使用哪种缓动 函数。
    - **complete**Type: Function()在动画完成时执行的函数。
    - **children** (default: `false`)Type: Boolean动画是否应用到所有匹配的元素的后代元素。应慎使用此功能。 因为要确定哪些后代元素要应用动画可能会非常好性能，而且 后代元素的数量是线性增长的。

- **queue** (default: `true`)Type: Boolean or String一个布尔值，指示是否将动画放置在效果队列中。如果为false时，将立即开始动画。 从**jQuery1.7**开始，队列选项也可以接受一个字符串，在这种情况下， 在动画被添加到由该字符串表示的队列中。

`.switchClass()` 方法允许你在展现动画的同时添加或移除样式类。

类似原生CSS transitions（过渡）， jQuery UI的样式动画提供了一个从一个状态到另一个状态平滑过渡， 同时让你 保持那些CSS样式改版的所有细节，和你的javaScript分离。 所有样式动画的方法，包括 `.switchClass()` 支持自定义的持续时间（durations）和缓冲函数（easing），以及在动画完成时提供一个回调。

并非所有的样式都可以设置动画。 例如，没有办法让背景图像应用动画。不能进行动画的样式，将在动画结束时被改变。

这个插件扩展了jQuery中的 `.switchClass()` 方法。 如果没有加载 jQuery UI，调用 `.switchClass()` 方法可能不会失败，因为方法仍然存在。 然而，预期的行为（愚人码头注：指平滑过渡动画效果）将不会发生。

Example:

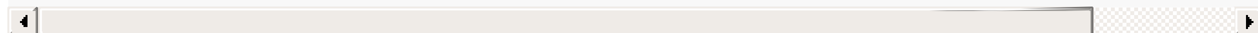
向匹配的元素添加样式类**"blue"**，并且移除样式类**"big"**。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>switchClass demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  div {
    width: 100px;
    height: 100px;
    background-color: #ccc;
  }
  .big {
    width: 200px;
    height: 200px;
  }
  .blue {
    background-color: #00f;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<div class="big"></div>

<script>
$( "div" ).click(function() {
  $( this ).switchClass( "big", "blue", 1000, "easeInOutQuad" );
});
</script>

</body>
</html>
```



.toggle()

Categories: Effects | Effects Core | Method Overrides | Methods

.toggle(effect [, options] [, duration] [, complete])Returns: jQuery

Description: 使用自定义特效显示或隐藏匹配的元素。

- ### .toggle(effect [, options] [, duration] [, complete])
  - **effect**Type: String一个字符串，指示哪个特效被使用。
  - **options**Type: Object特效选项和缓冲函数。
  - **duration** (default: `400`)Type: Number or String一个字符串或者数字决定动画将运行多久。（愚人码头注：三种预定速度的字符串("slow", "normal", 或 "fast")或表示动画时长的毫秒数值(如：1000)）
  - **complete**Type: Function()在动画完成时执行的函数。

- ### .toggle(options)
  - **options**Type: Object全部动画选项。 只有一个必须设置的属性 `effect` 。
    - **effect**Type: String一个字符串，指示哪个特效被使用。
    - **easing** (default: `"swing"`)Type: String一个字符串，表示过渡使用哪种 缓动 函数。
    - **duration** (default: `400`)Type: Number or String一个字符串或者数字决定动画将运行多久。（愚人码头注：三种预定速度的字符串("slow", "normal", 或 "fast")或表示动画时长的毫秒数值(如：1000)）
    - **complete**Type: Function()在动画完成时执行的函数。
    - **queue** (default: `true`)Type: Boolean or String一个布尔值，指示是否将动画放置在效果队列中。如果为false时，将立即开始动画。 从**jQuery1.7**开始，队列选项也可以接受一个字符串，在这种情况下， 在动画被添加到由该字符串表示的队列中。

这个插件扩展了jQuery中的 `.toggle()` 方法。 如果没有加载 jQuery UI，调用 `.toggle()` 方法可能不会失败，因为方法仍然存在。 然而，预期的行为（愚人码头注：指平滑过渡动画效果）将不会发生。

Example:

使用折叠特效来切换显示一个div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>toggle demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  div {
    width: 100px;
    height: 100px;
    background: #ccc;
    border: 1px solid #000;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<button>toggle the div</button>
<div></div>

<script>
$( "button" ).click(function() {
  $( "div" ).toggle( "fold", 1000 );
});
</script>

</body>
</html>
```

.toggleClass()

Categories: Effects | Effects Core | Method Overrides

.toggleClass(className [, switch] [, duration] [, easing] [, complete])Returns: jQuery

Description: 为每一组匹配的元素添加或移除一个或多个样式类，取决于当前元素是否存在该样式类和switch参数。，而且所有改变的样式以动画的形式展示

- **.toggleClass(className [, switch] [, duration] [, easing] [, complete])**
  - **className**Type: String将要被添加到匹配的元素的class属性的一个或多个名称(以空格分开)。
  - **switch**Type: Boolean一个决定是否添加或移除某css类的布尔值。
  - **duration** (default: `400`)Type: Number or String一个字符串或者数字决定动画将运行多久。（愚人码头注：三种预定速度的字符串("slow", "normal", 或 "fast")或表示动画时长的毫秒数值(如：1000) ）
  - **easing** (default: `swing`)Type: String一个字符串，表示过渡使用哪种 缓动 函数。
  - **complete**Type: Function()在动画完成时执行的函数。
- **.toggleClass(className [, switch] [, options])**
  - **className**Type: String将要被添加到匹配的元素的class属性的一个或多个名称(以空格分开)。
  - **switch**Type: Boolean一个决定是否添加或移除某css类的布尔值。
  - **options**Type: Object一组包含动画选项的值的集合。所有属性是可选择的。
    - **duration** (default: `400`)Type: Number or String一个字符串或者数字决定动画将运行多久。（译者注：默认值: "normal"， 三种预定速度的字符串("slow", "normal", 或 "fast")或表示动画时长的毫秒数值(如：1000) ）
    - **easing** (default: `swing`)Type: String一个字符串，表示过渡使用哪种缓动 函数。
    - **complete**Type: Function()在动画完成时执行的函数。
    - **children** (default: `false`)Type: Boolean动画是否应用到所有匹配的元素的后代元素。应慎使用此功能。 因为要确定哪些后代元素要应用动画可能会非常好性能，而且 后代元素的数量是线性增长的。
    - **queue** (default: `true`)Type: Boolean or String一个布尔值，指示是否将动画放置在效果队列中。如果为false时，将立即开始动画。 从**jQuery1.7**开始，队列选项也可以接受一个字符串，在这种情况下， 在动画被添加到由该字符串表示

的队列中。

类似原生CSS transitions（过渡）， jQuery UI的样式动画提供了一个从一个状态到另一个状态平滑过渡， 同时让你 保持那些CSS样式改版的所有细节，和你的javaScript分离。 所有样式动画的方法，包括 `.toggleClass()` 支持自定义的持续时间（durations）和缓冲函数（easing），以及在动画完成时提供一个回调。

并非所有的样式都可以设置动画。 例如，没有办法让背景图像应用动画。不能进行动画的样式，将在动画结束时被改变。

这个插件扩展了jQuery中的 `.toggleClass()` 方法。 如果没有加载 jQuery UI，调用 `.toggleClass()` 方法可能不会失败，因为方法仍然存在。 然而，预期的行为（愚人码头注：指平滑过渡动画效果）将不会发生。

Example:

把类**"big-blue"**添加到匹配的元素

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>toggleClass demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  div {
    width: 100px;
    height: 100px;
    background-color: #ccc;
  }
  .big-blue {
    width: 200px;
    height: 200px;
    background-color: #00f;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<div></div>

<script>
$( "div" ).click(function() {
  $( this ).toggleClass( "big-blue", 1000, "easeOutSine" );
});
</script>

</body>
</html>
```

Transfer Effect

Categories: Effects

Description: Transfers the outline of an element to another element

- **transfer**
  - **className**Type: String传递给目标元素的样式类名。
  - **to**Type: StringjQuery选择器, 要转移到的目标元素。

当试图在两个元素之间呈现互动效果时，该特效非常有用。

被转移元素自己有一个样式类 `ui-effects-transfer` ，但此类需要开发者自己配置，例如添加背景颜色或边框。

Example:

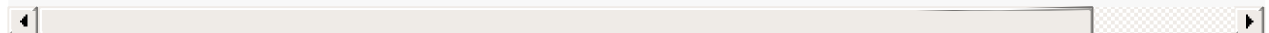
点击绿色元素将其轮廓转移到另一个元素上。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>transfer demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  div.green {
    width: 100px;
    height: 80px;
    background: green;
    border: 1px solid black;
    position: relative;
  }
  div.red {
    margin-top: 10px;
    width: 50px;
    height: 30px;
    background: red;
    border: 1px solid black;
    position: relative;
  }
  .ui-effects-transfer {
    border: 1px dotted black;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<div class="green"></div>
<div class="red"></div>

<script>
$( "div" ).click(function() {
  var i = 1 - $( "div" ).index( this );
  $( this ).effect( "transfer", { to: $( "div" ).eq( i ) }, 1000 );
});
</script>

</body>
</html>
```



Effect Core

由 jquery.ui.effect.js 提供的功能。除了下面列出的方法，jquery.ui.effect.js 还包括一些 Easings。

Also in: Effects | Method Overrides

.addClass()

当动画样式改变时，为匹配的元素集合内的每个元素添加指定的 Class。

Color Animation

使用 .animate() 实现颜色动画效果。

Also in: Effects | Methods

.effect()

对一个元素应用动画特效。

Also in: Effects | Method Overrides | Methods

.hide()

使用自定义效果来隐藏匹配的元素。

Also in: Effects | Method Overrides

.removeClass()

当动画样式改变时，为匹配的元素集合内的每个元素移除指定的 Class。

Also in: Effects | Method Overrides | Methods

.show()

使用自定义效果来显示匹配的元素。

Also in: Effects

.switchClass()

当动画样式改变时，为匹配的元素集合内的每个元素添加和移除指定的 Class。

Also in: Effects | Method Overrides | Methods

.toggle()

使用自定义效果来显示或隐藏匹配的元素。

Also in: Effects | Method Overrides

.toggleClass()

当动画样式改变时，根据 Class 是否存在以及 switch 参数的值，为匹配的元素集合内的每个元素添加或移除一个或多个 Class。

Interactions

jQuery UI 提供了一套基于鼠标的交互。

Draggable Widget

允许使用鼠标移动元素。

Droppable Widget

为可拖拽小部件创建目标。

Also in: Utilities

Mouse Interaction

基本交互层。

Resizable Widget

使用鼠标改变元素的尺寸。

Selectable Widget

使用鼠标选择单个元素或一组元素。

Sortable Widget

使用鼠标调整列表中或者网格中元素的排序。

Draggable Widget

Categories: Interactions

version added: 1.0

Description: 该组件可以让你使用鼠标拖动所有元素。

QuickNavExamples

Options

- addClasses
- appendTo
- axis
- cancel
- connectToSortable
- containment
- cursor
- cursorAt
- delay
- disabled
- distance
- grid
- handle
- helper
- iframeFix
- opacity
- refreshPositions
- revert
- revertDuration
- scope
- scroll
- scrollSensitivity
- scrollSpeed
- snap
- snapMode

- snapTolerance
- stack
- zIndex

Methods

- destroy
- disable
- enable
- option
- widget

Events

- create
- drag
- start
- stop

让被选择元素可以被鼠标拖动。如果你想把元素拖放到另一个元素内部，查看jQuery UI Droppable plugin,该组件为被拖动元素提供了一个目标容器。

Dependencies（依赖性）

- UI Core
- Widget Factory
- Mouse Interaction

Options

addClassesType: Boolean

Default: `true` 如果值设置为 `false`，`ui-draggable` 样式类将不能被添加引用。当在大量元素上调用 `.draggable()` 时，你可能会想要这样设置，作为一个性能优化。**Code examples:**

使用 `addClasses` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ addClasses: false });
```

在组件初始化之后，读取或设置 `addClasses` 选项

```
// getter
var addClasses = $( ".selector" ).draggable( "option", "addClasses" );

// setter
$( ".selector" ).draggable( "option", "addClasses", false );
```

appendToType: jQuery or Element or Selector or String

Default: `"parent"` 当进行拖动时，拖动组件助手应该被添加到的元素。支持多种类型:

- **jQuery**: 一个含有要被添加拖动组件助手的元素的Jquery对象。
- **Element**: 要被添加拖动组件助手的元素。
- **Selector**: 一个用来识别要被添加拖动组件助手的元素的选择器。
- **String**: 字符串 `"parent"` 将会使拖动组件助手成为组件的同级元素。

Code examples:

使用 `appendTo` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ appendTo: "body" });
```

在组件初始化之后，读取或设置 `appendTo` 选项

```
// getter
var appendTo = $( ".selector" ).draggable( "option", "appendTo" );

// setter
$( ".selector" ).draggable( "option", "appendTo", "body" );
```

axisType: String

Default: `false` 约束拖动的动作只能在水平(x轴)或垂直(y轴)上执行。可选值: `"x"` , `"y"` 。**Code examples:**

使用 `axis` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ axis: "x" });
```

在组件初始化之后，读取或设置 `axis` 选项：

```
// getter
var axis = $( ".selector" ).draggable( "option", "axis" );

// setter
$( ".selector" ).draggable( "option", "axis", "x" );
```

cancelType: Selector

Default: `"input,textarea,button,select,option"` 防止在指定元素上开始拖动。**Code examples:**

使用 `cancel` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ cancel: ".title" });
```

在组件初始化之后，读取或设置 `cancel` 选项：

```
// getter
var cancel = $( ".selector" ).draggable( "option", "cancel" );

// setter
$( ".selector" ).draggable( "option", "cancel", ".title" );
```

connectToSortableType: Selector

Default: `false` 允许draggable被拖拽到指定的sortables中。如果使用了该选项，被拖动的元素可以被放置于一个应用了排序组件的元素列表中并成为该列表的一部分。注意: 为了完美的使用该特性， `helper` 选项必须被设置为 `"clone"` 。 必须包含jQuery UI 排序组件。**Code examples:**

使用 `connectToSortable` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ connectToSortable: "#my-sortable" });
```

在组件初始化之后，读取或设置 `connectToSortable` 选项：

```
// getter
var connectToSortable = $( ".selector" ).draggable( "option", "connectToSortable" );

// setter
$( ".selector" ).draggable( "option", "connectToSortable", "#my-sortable" );
```

containmentType: Selector or Element or String or Array

Default: `false` 可以限制draggable只能在指定的元素或区域的边界以内进行拖动。支持多种类型:

- **Selector**: 可拖动元素将被置于由选择器指定的第一个元素的起界限作用的盒模型中。如果没有找到任何元素，则不会设置界限。
- **Element**: 可拖动的元素将包含该元素的边界框。
- **String**:可选值: `"parent"` , `"document"` , `"window"` .
- **Array**: 以 `[ x1, y1, x2, y2 ]` 数组形式定义一个限制区域.

Code examples:

使用 `containment` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ containment: "parent" });
```

在组件初始化之后，读取或设置 `containment` 选项：

```
// getter
var containment = $( ".selector" ).draggable( "option", "containment" );

// setter
$( ".selector" ).draggable( "option", "containment", "parent" );
```

cursorType: String

Default: `"auto"` 指定在做拖拽动作时，鼠标的CSS样式。**Code examples:**

使用 `cursor` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ cursor: "crosshair" });
```

在组件初始化之后，读取或设置 `cursor` 选项：

```
// getter
var cursor = $( ".selector" ).draggable( "option", "cursor" );

// setter
$( ".selector" ).draggable( "option", "cursor", "crosshair" );
```

cursorAtType: Object

Default: `false` 设置拖动帮手相对于鼠标光标的偏移量。坐标可被指定为一个散列 使用的组合中的一个或两个键： `{ top, left, right, bottom }` 。**Code examples:**

使用 `cursorAt` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ cursorAt: { left: 5 } });
```

在组件初始化之后，读取或设置 `cursorAt` 选项：

```
// getter
var cursorAt = $( ".selector" ).draggable( "option", "cursorAt" );

// setter
$( ".selector" ).draggable( "option", "cursorAt", { left: 5 } );
```

delayType: Number

Default: `0` 当鼠标按下后，指定时延迟间后才开始激活拖拽动作（单位：毫秒）。此选项可用来阻止当点击一个元素时可能发生的非期望拖动行为。**Code examples:**

使用 `delay` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ delay: 300 });
```

在组件初始化之后，读取或设置 `delay` 选项：

```
// getter
var delay = $( ".selector" ).draggable( "option", "delay" );

// setter
$( ".selector" ).draggable( "option", "delay", 300 );
```

disabledType: Boolean

Default: `false` 如果该值设置为 `true`，拖动特效将被禁用。**Code examples:**

使用 `disabled` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ disabled: true });
```

在组件初始化之后，读取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).draggable( "option", "disabled" );

// setter
$( ".selector" ).draggable( "option", "disabled", true );
```

distanceType: Number

Default: `1` 当鼠标点下后，只有移动指定像素后才开始激活拖拽动作，单位为像素。此选项可用来阻止当点击一个元素时可能发生的非期望拖动行为。**Code examples:**

使用 `distance` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ distance: 10 });
```

在组件初始化之后，读取或设置 `distance` 选项：

```
// getter
var distance = $( ".selector" ).draggable( "option", "distance" );

// setter
$( ".selector" ).draggable( "option", "distance", 10 );
```

gridType: Array

Default: `false` 拖拽元素时，只能以指定大小的方格进行拖动。数组形式为 `[ x, y ]` 。**Code examples:**

使用 `grid` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ grid: [ 50, 20 ] });
```

在组件初始化之后，读取或设置 `grid` 选项:

```
// getter
var grid = $( ".selector" ).draggable( "option", "grid" );

// setter
$( ".selector" ).draggable( "option", "grid", [ 50, 20 ] );
```

handleType: Selector or Element

Default: `false` 当参数值为true时,只有在特定的元素上触发鼠标按下事件时，才可以拖动。只有被拖到元素的子元素才可以应用该参数。**Code examples:**

使用 `handle` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ handle: "h2" });
```

在组件初始化之后，读取或设置 `handle` 选项:

```
// getter
var handle = $( ".selector" ).draggable( "option", "handle" );

// setter
$( ".selector" ).draggable( "option", "handle", "h2" );
```

helperType: String or Function()

Default: `"original"` 允许一个元素被用来进行拖动。支持多种类型:

- **String**:如果值设置为 `"clone"` ，那么该元素将会被复制，并且被复制的元素将被拖动。
- **Function**: 当拖动时， 函数将返回一个DOM元素以供使用。

Code examples:

使用 `helper` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ helper: "clone" });
```

在组件初始化之后，读取或设置 `helper` 选项：

```
// getter
var helper = $( ".selector" ).draggable( "option", "helper" );

// setter
$( ".selector" ).draggable( "option", "helper", "clone" );
```

iframeFixType: Boolean or Selector

Default: `false` 在拖动期间阻止iframe捕捉鼠标移过事件。与 `cursorAt` 选项搭配使用时或者当鼠标指针可能不在拖动助手元素之上时，该参数非常有用。支持多种类型:

- **Boolean**: 当设置为 `true` , 透明层将被放置于页面上的所有iframe之上。/li>
- **Selector**: 任何由选择器匹配的iframe将被透明层覆盖。

Code examples:

使用 `iframeFix` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ iframeFix: true });
```

在组件初始化之后，读取或设置 `iframeFix` 选项：

```
// getter
var iframeFix = $( ".selector" ).draggable( "option", "iframeFix" );

// setter
$( ".selector" ).draggable( "option", "iframeFix", true );
```

opacityType: Number

Default: `false` 当拖动时，拖动助手元素的不透明度。**Code examples:**

使用 `opacity` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ opacity: 0.35 });
```

在组件初始化之后，读取或设置 `opacity` 选项：

```
// getter
var opacity = $( ".selector" ).draggable( "option", "opacity" );

// setter
$( ".selector" ).draggable( "option", "opacity", 0.35 );
```

refreshPositionsType: Boolean

Default: `false` 如果设置为 `true` ， 所有的可拖动位置在每次鼠标移动时都会被计算。 注意: 这解决了具有高度动态内容页面的问题，但是却降低了性能。**Code examples:**

使用 `refreshPositions` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ refreshPositions: true });
```

在组件初始化之后，读取或设置 `refreshPositions` 选项:

```
// getter
var refreshPositions = $( ".selector" ).draggable( "option", "refreshPositions" );

// setter
$( ".selector" ).draggable( "option", "refreshPositions", true );
```

revertType: Boolean or String

Default: `false` 当拖动停止时，元素是否要被重置到它的初始位置。支持多种类型:

- **Boolean**: 如果设置为 `true` ， 当拖动停止时，元素位置将被重置。
- **String**: 如果设置为 `"invalid"` , 重置仅当拖动没有被放置于一个可放置的对象时才会发生。如果设置为 `"valid"` , 情况则相反。

Code examples:

使用 `revert` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ revert: true });
```

在组件初始化之后，读取或设置 `revert` 选项:

```
// getter
var revert = $( ".selector" ).draggable( "option", "revert" );

// setter
$( ".selector" ).draggable( "option", "revert", true );
```

revertDurationType: Number

Default: `500` 动画重置时间, 单位为微秒。如果 `revert` 选项设置为 `false` ， 则忽略（译者注：即被拖到元素不会被重置。）。**Code examples:**

使用 `revertDuration` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ revertDuration: 200 });
```

在组件初始化之后，读取或设置 `revertDuration` 选项：

```
// getter
var revertDuration = $( ".selector" ).draggable( "option", "revertDuration" );

// setter
$( ".selector" ).draggable( "option", "revertDuration", 200 );
```

scopeType: String

Default: `"default"` 被用来将拖动组件和拖动至容器组件的参数进行分组, 除了拖动至容器组件的 `accept` 选项。如果一个一般拖动组件的 `scope` 参数值和一个拖动至容器组件的 `scope` 参数值一样，那么这个一般拖动组件将被接受为拖动至容器组件。**Code examples:**

使用 `scope` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ scope: "tasks" });
```

在组件初始化之后，读取或设置 `scope` 选项：

```
// getter
var scope = $( ".selector" ).draggable( "option", "scope" );

// setter
$( ".selector" ).draggable( "option", "scope", "tasks" );
```

scrollType: Boolean

Default: `true` 如果设置为 `true` , 当拖动时，div盒模型将自动翻滚。**Code examples:**

使用 `scroll` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ scroll: false });
```

在组件初始化之后，读取或设置 `scroll` 选项：

```
// getter
var scroll = $( ".selector" ).draggable( "option", "scroll" );

// setter
$( ".selector" ).draggable( "option", "scroll", false );
```

scrollSensitivityType: Number

Default: `20` 离开可视区域边缘多少距离开始滚动。距离是相对指针进行计算的，而不是被拖到元素本身。如果 `scroll` 选项设置为 `false` ，则不滚动。**Code examples:**

使用 `scrollSensitivity` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ scrollSensitivity: 100 });
```

在组件初始化之后，读取或设置 `scrollSensitivity` 选项:

```
// getter
var scrollSensitivity = $( ".selector" ).draggable( "option", "scrollSensitivity" );

// setter
$( ".selector" ).draggable( "option", "scrollSensitivity", 100 );
```

scrollSpeedType: Number

Default: `20` 当鼠标指针的值小于 `scrollSensitivity` 的值时，窗口滚动的速度。如果 `scroll` 选项设置为 `false` ，则该参数无效。**Code examples:**

使用 `scrollSpeed` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ scrollSpeed: 100 });
```

在组件初始化之后，读取或设置 `scrollSpeed` 选项:

```
// getter
var scrollSpeed = $( ".selector" ).draggable( "option", "scrollSpeed" );

// setter
$( ".selector" ).draggable( "option", "scrollSpeed", 100 );
```

snapType: Boolean or Selector

Default: `false` 决定一个元素是否应该吸附到其它元素上。支持多种类型:

- **Boolean**: 当设置为 `true` 时, 元素将可以吸附到所有其它可拖动元素上。
- **Selector**: 确定被吸附元素。

Code examples:

使用 `snap` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ snap: true });
```

在组件初始化之后，读取或设置 `snap` 选项:

```
// getter
var snap = $( ".selector" ).draggable( "option", "snap" );

// setter
$( ".selector" ).draggable( "option", "snap", true );
```

snapModeType: String

Default: `"both"` 决定可拖动元素将要吸附到哪个元素的边缘。如果 `snap` 选项设置为 `false` ，则忽略该参数。 可选值: `"inner"` , `"outer"` , `"both"` .**Code examples:**

使用 `snapMode` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ snapMode: "inner" });
```

在组件初始化之后，读取或设置 `snapMode` 选项:

```
// getter
var snapMode = $( ".selector" ).draggable( "option", "snapMode" );

// setter
$( ".selector" ).draggable( "option", "snapMode", "inner" );
```

snapToleranceType: Number

Default: `20` 当距离可吸附元素多远时，触发吸附事件。如果 `snap` 选项设置为 `false` ，则忽略该参数。**Code examples:**

使用 `snapTolerance` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ snapTolerance: 30 });
```

在组件初始化之后，读取或设置 `snapTolerance` 选项:

```
// getter
var snapTolerance = $( ".selector" ).draggable( "option", "snapTolerance" );

// setter
$( ".selector" ).draggable( "option", "snapTolerance", 30 );
```

stackType: Selector

Default: `false` 控制由选择器所触发的一系列元素的z-index值，总是将当前被拖动对象至于最前。对于像窗口管理这样的应用来说，非常有用。**Code examples:**

使用 `stack` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ stack: ".products" });
```

在组件初始化之后，读取或设置 `stack` 选项:

```
// getter
var stack = $( ".selector" ).draggable( "option", "stack" );

// setter
$( ".selector" ).draggable( "option", "stack", ".products" );
```

zIndexType: Number

Default: `false` 当被拖动时，拖动助手的Z-index值。**Code examples:**

使用 `zIndex` 选项初始化Draggable Widget：

```
$( ".selector" ).draggable({ zIndex: 100 });
```

在组件初始化之后，读取或设置 `zIndex` 选项:

```
// getter
var zIndex = $( ".selector" ).draggable( "option", "zIndex" );

// setter
$( ".selector" ).draggable( "option", "zIndex", 100 );
```

Methods

destroy()

完全销毁一般拖动组件的功能，这将使元素返回它的初始状态。

- 这个方法不接受任何参数。

Code examples:

请求destroy方法:

```
$( ".selector" ).draggable( "destroy" );
```

disable()

禁用一般拖动组件。

- 这个方法不接受任何参数。

Code examples:

请求disable方法:

```
$( ".selector" ).draggable( "disable" );
```

enable()

启用一般拖动组件。

- 这个方法不接受任何参数。

Code examples:

请求enable方法:

```
$( ".selector" ).draggable( "enable" );
```

option(optionName)Returns: Object

获取与 `optionName` 对应的参数值。

- **optionName**Type: String要获取的值所对应的选项的名称。

Code examples:

请求方法:

```
var isDisabled = $( ".selector" ).draggable( "option", "disabled" );
```

option()Returns: PlainObject

获取一个包含有描述当前一般拖动组件选项哈希值的键/值对。

- 这个方法不接受任何参数。

Code examples:

请求方法:

```
var options = $( ".selector" ).draggable( "option" );
```

option(optionName, value)

设置一般拖动组件选项的值，选项名称由 `optionName` 指定。

- **optionName**Type: String要设置的选项的名称。
- **value**Type: Object要为选项设置的参数值。

Code examples:

请求方法:

```
$( ".selector" ).draggable( "option", "disabled", true );
```

option(options)

为一般拖动组件设置一个或多个选项值。

- **options**Type: Object要设置的选项与值之间的映射关系。

Code examples:

请求方法:

```
$( ".selector" ).draggable( "option", { disabled: true } );
```

widget()Returns: jQuery

返回一个包含有可拖动元素的 `jQuery` 对象。

- 这个方法不接受任何参数。

Code examples:

请求widget方法:

```
var widget = $( ".selector" ).draggable( "widget" );
```

Events

create(event, ui)Type: `dragcreate`

当一般拖动组件被创建时触发。

- **event**Type: Event
- **ui**Type: Object

注意: *`ui`* 对象是空的，但是为了与其它元素保持一直，它总是被包含。

Code examples:

使用create回调函数指定事件:

```
$( ".selector" ).draggable({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到dragcreate事件:

```
$( ".selector" ).on( "dragcreate", function( event, ui ) {} );
```

drag(event, ui)Type: `drag`

当鼠标在拖动过程中移动时触发。

- **event**Type: Event
- **ui**Type: Object
    - **helper**Type: jQuery代表被拖动的元素。
    - **position**Type: Object当前元素的CSS位置，以 `{ top, left }` 形式给出。
    - **offset**Type: Object当前元素的偏移位置，以 `{ top, left }` 形式给出。

Code examples:

使用drag回调函数指定事件:

```
$( ".selector" ).draggable({
  drag: function( event, ui ) {}
});
```

绑定一个事件监听者到drag事件:

```
$( ".selector" ).on( "drag", function( event, ui ) {} );
```

start(event, ui)Type: `dragstart`

当拖动开始时触发。

- **event**Type: Event
- **ui**Type: Object
    - **helper**Type: jQuery代表被拖动的元素。
    - **position**Type: Object当前元素的CSS位置，以 `{ top, left }` 形式给出。
    - **offset**Type: Object当前元素的偏移位置，以 `{ top, left }` 形式给出。

Code examples:

使用start callback specified:

```
$( ".selector" ).draggable({
  start: function( event, ui ) {}
});
```

拖动事件绑定一个事件监听器：

```
$( ".selector" ).on( "dragstart", function( event, ui ) {} );
```

stop(event, ui)Type: `dragstop`

当拖动停止时触发。

- **event**Type: Event
- **ui**Type: Object
  - **helper**Type: jQuery代表被拖动的元素。
  - **position**Type: Object当前元素的CSS位置，以 `{ top, left }` 形式给出。
  - **offset**Type: Object当前元素的偏移位置，以 `{ top, left }` 形式给出。

Code examples:

使用start回调函数指定事件:

```
$( ".selector" ).draggable({
  stop: function( event, ui ) {}
});
```

绑定一个事件监听者到dragstart事件:

```
$( ".selector" ).on( "dragstop", function( event, ui ) {} );
```

Example:

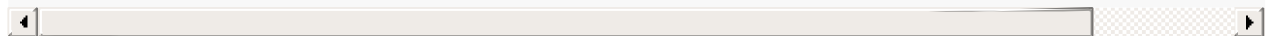
一个简单的jQuery UI一般拖动

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>draggable demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  #draggable {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<div id="draggable">Drag me</div>

<script>
$( "#draggable" ).draggable();
</script>

</body>
</html>
```



Droppable Widget

Categories: Interactions

version added: 1.0

Description: 创建拖动元素的目标。

QuickNavExamples

Options

- accept
- activeClass
- addClasses
- disabled
- greedy
- hoverClass
- scope
- tolerance

Methods

- destroy
- disable
- enable
- option
- widget

Events

- activate
- create
- deactivate
- drop
- out
- over

jQuery UI拖放插件可以使所选元素可拖放（这意味着draggables拖动可以被拖放接受）。您可以指定哪个拖动被接受。

Dependencies（依赖性）

- UI Core
- Widget Factory
- Mouse Interaction

Options

acceptType: Selector or Function()

Default: `"*"` 控制可拖动的元素被拖放接受。支持多种类型：

- **Selector**: 一个选择器可拖动的元素。
- **Function**: 函数将被调用页面上的每一个可拖动的（传递给函数的第一个参数）。该函数必须返回 `true`， 如果可拖动应该接受。

Code examples:

使用指定的 `accept` 参数初始化一个droppable：

```
$( ".selector" ).droppable({ accept: ".special" });
```

在初始化后设置或者获取 `accept` 参数：

```
// getter
var accept = $( ".selector" ).droppable( "option", "accept" );

// setter
$( ".selector" ).droppable( "option", "accept", ".special" );
```

activeClassType: String

Default: `false` 如果指定了该参数,在被允许的draggable对象填充时,droppable会被添加上指定的样式。**Code examples:**

使用指定的 `activeClass` 参数初始化一个droppable：

```
$( ".selector" ).droppable({ activeClass: "ui-state-highlight" });
```

在初始化后设置或者获取 `activeClass` 参数：

```
// getter
var activeClass = $( ".selector" ).droppable( "option", "activeClass" );

// setter
$( ".selector" ).droppable( "option", "activeClass", "ui-state-highlight" );
```

addClassesType: Boolean

Default: `true` 如果设置为 `false` , 可以防止 `ui-droppable` 类在进行时添加. 这可能会使在初始化数百个元素执行 `.droppable()` 时使性能得到理想的优化.**Code examples:**

使用指定的 `addClasses` 参数初始化一个droppable：

```
$( ".selector" ).droppable({ addClasses: false });
```

在初始化后设置或者获取 `addClasses` 参数：

```
// getter
var addClasses = $( ".selector" ).droppable( "option", "addClasses" );

// setter
$( ".selector" ).droppable( "option", "addClasses", false );
```

disabledType: Boolean

Default: `false` 如果设置为 `true` 将禁止拖放。**Code examples:**

使用指定的 `disabled` 参数初始化一个droppable：

```
$( ".selector" ).droppable({ disabled: true });
```

在初始化后设置或者获取 `disabled` 参数：

```
// getter
var disabled = $( ".selector" ).droppable( "option", "disabled" );

// setter
$( ".selector" ).droppable( "option", "disabled", true );
```

greedyType: Boolean

Default: `false` 默认情况下，当一个元素被放到嵌套的放置（droppable）对象时，每个放置（droppable）对象都将接收到这个元素。然而，当设置这个选项为 `true` 时，任何父级的放置（droppable）对象不会接收元素。 `drop` 事件将依然会正常的泡沫，但 `event.target` 查看哪个放置（droppable）对象接受了拖动元素。**Code examples:**

使用指定的 `greedy` 参数初始化一个droppable：

```
$( ".selector" ).droppable({ greedy: true });
```

在初始化后设置或者获取 `greedy` 参数：

```
// getter
var greedy = $( ".selector" ).droppable( "option", "greedy" );

// setter
$( ".selector" ).droppable( "option", "greedy", true );
```

hoverClassType: String

Default: `false` 如果设置了该参数,将在一个被允许的拖动元素悬停在放置（droppable）对象上时，向放置（droppable）对象添加一个指定的样式.**Code examples:**

使用指定的 `hoverClass` 参数初始化一个droppable：

```
$( ".selector" ).droppable({ hoverClass: "drop-hover" });
```

在初始化后设置或者获取 `hoverClass` 参数：

```
// getter
var hoverClass = $( ".selector" ).droppable( "option", "hoverClass" );

// setter
$( ".selector" ).droppable( "option", "hoverClass", "drop-hover" );
```

scopeType: String

Default: `"default"` 用来设置拖动（draggle）元素和放置（droppable）对象的集合, 除了droppable中的 `accept` 属性指定的元素外,和放置（droppable）对象相同集合的放置（droppable）对象也被允许添加到放置（droppable）对象中.**Code examples:**

使用指定的 `scope` 参数初始化一个droppable：

```
$( ".selector" ).droppable({ scope: "tasks" });
```

在初始化后设置或者获取 `scope` 参数：

```
// getter
var scope = $( ".selector" ).droppable( "option", "scope" );

// setter
$( ".selector" ).droppable( "option", "scope", "tasks" );
```

toleranceType: String

Default: `"intersect"` 指定使用那种模式来测试一个拖动(draggable)元素"经过"一个放置（droppable）对象。 允许使用的值:

- `"fit"` : 拖动(draggable)元素 完全重叠到放置（droppable）对象。
- `"intersect"` : 拖动(draggable)元素 和放置（droppable）对象至少重叠50%。
- `"pointer"` : 鼠标重叠到放置（droppable）对象上。
- `"touch"` : 拖动(draggable)元素 和放置（droppable）对象的任意重叠

Code examples:

使用指定的 `tolerance` 参数初始化一个droppable：

```
$( ".selector" ).droppable({ tolerance: "fit" });
```

在初始化后设置或者获取 `tolerance` 参数：

```
// getter
var tolerance = $( ".selector" ).droppable( "option", "tolerance" );

// setter
$( ".selector" ).droppable( "option", "tolerance", "fit" );
```

Methods（方法）

destroy()

完全移除拖放功能. 这将使元素返回到之前的初始化状态.

- 这个方法不接受任何参数

Code examples:

调用destroy 方法

```
$( ".selector" ).droppable( "destroy" );
```

disable()

禁用拖放。

- 这个方法不接受任何参数

Code examples:

调用disable 方法

```
$( ".selector" ).droppable( "disable" );
```

enable()

启用拖放。

- 这个方法不接受任何参数

Code examples:

调用enable 方法

```
$( ".selector" ).droppable( "enable" );
```

option(optionName)Returns: Object

通过指定的 `optionName` 获取当前关联的值。

- **optionName**Type: String要获取值的选项名

Code examples:

调用这个方法：

```
var isDisabled = $( ".selector" ).droppable( "option", "disabled" );
```

option()Returns: PlainObject

获取一个对象，它包含表示当前droppable的选项hash的键/值对。

- 这个方法不接受任何参数

Code examples:

调用这个方法：

```
var options = $( ".selector" ).droppable( "option" );
```

option(optionName, value)

通过指定的 `optionName` ，设置droppable的相关选项值。

- **optionName**Type: String要设置值的选项名。

- **value**Type: Object要设置选项的值。

Code examples:

调用这个方法：

```
$( ".selector" ).droppable( "option", "disabled", true );
```

option(options)

为droppable设置一个或多个选项。

- **options**Type: Object设置的选项/值对的对象。

Code examples:

调用这个方法：

```
$( ".selector" ).droppable( "option", { disabled: true } );
```

widget()Returns: jQuery

返回一个 `jQuery` ，它包含了droppable元素。

- 这个方法不接受任何参数

Code examples:

调用widget 方法

```
var widget = $( ".selector" ).droppable( "widget" );
```

Events

activate(event, ui)Type: `dropactivate`

这个事件会在任何允许的draggable对象开始拖动时触发。它可以用来在你想让droppable对象在可以被填充的时"亮起来"的时候。

- **event**Type: Event
- **ui**Type: Object
  - **draggable**Type: jQuery一个jQuery对象代表的拖动元素。
  - **helper**Type: jQuery一个jQuery对象代表被拖动元素的助手。
  - **position**Type: Object当前可拖动助手的CSS的position（位置）对象，

如 `{ top, left }` 。
  - **offset**Type: Object当前可拖动助手的偏移位置对象，如 `{ top, left }` 。

Code examples:

使用指定的activate 回调初始化一个droppable：

```
$( ".selector" ).droppable({
 activate: function( event, ui ) {}
});
```

绑定一个事件监听到dropactivate事件:

```
$( ".selector" ).on( "dropactivate", function( event, ui ) {} );
```

create(event, ui)Type: `dropcreate`

此事件会在droppable创建时触发。

- **event**Type: Event
- **ui**Type: Object

注意：`ui` 对象是空对象，包括是为了和其他事件的一致性。

Code examples:

使用指定的create 回调初始化一个droppable：

```
$( ".selector" ).droppable({
 create: function( event, ui ) {}
});
```

绑定一个事件监听到dropcreate事件：

```
$( ".selector" ).on( "dropcreate", function( event, ui ) {} );
```

deactivate(event, ui)Type: `dropdeactivate`

此事件会在任何允许的draggable对象停止拖动时触发.

- **event**Type: Event
- **ui**Type: Object
  - **draggable**Type: jQuery一个jQuery对象代表的拖动元素。
  - **helper**Type: jQuery一个jQuery对象代表被拖动元素的助手。
  - **position**Type: Object当前可拖动助手的CSS的position（位置）对象，

  如 `{ top, left }`。
- **offset**Type: Object当前可拖动助手的偏移位置对象，如 `{ top, left }`。

Code examples:

使用指定的deactivate 回调初始化一个droppable：

```
$( ".selector" ).droppable({
 deactivate: function( event, ui ) {}
});
```

绑定一个事件监听到dropdeactivate事件：

```
$( ".selector" ).on( "dropdeactivate", function( event, ui ) {} );
```

drop(event, ui)Type: `drop`

这个事件会在一个允许的拖动（draggable）元素填充进这个放置（droppable）对象时触发.（基于 `tolerance` 选项）。（愚人码头注释： 回调函数中, $(this) 表示被填充的droppable对象. ui.draggable 表示draggable对象.）

- **event**Type: Event
- **ui**Type: Object
  - **draggable**Type: jQuery一个jQuery对象代表的拖动元素。
  - **helper**Type: jQuery一个jQuery对象代表被拖动元素的助手。
  - **position**Type: Object当前可拖动助手的CSS的position（位置）对象，如 `{ top, left }`。
  - **offset**Type: Object当前可拖动助手的偏移位置对象，如 `{ top, left }`。

Code examples:

使用指定的drop 回调初始化一个droppable：

```
$( ".selector" ).droppable({
 drop: function( event, ui ) {}
});
```

绑定一个事件监听到drop事件：

```
$( ".selector" ).on( "drop", function( event, ui ) {} );
```

out(event, ui)Type: `dropout`

此事件会在一个允许的拖动（draggable）元素离开这个放置（droppable）对象时触发（基于 `tolerance` 选项）。

- **event**Type: Event
- **ui**Type: Object

注意：`ui` 对象是空对象，包括是为了和其他事件的一致性。

Code examples:

使用指定的out 回调初始化一个droppable：

```
$( ".selector" ).droppable({
 out: function( event, ui ) {}
});
```

绑定一个事件监听到dropout事件：

```
$( ".selector" ).on( "dropout", function( event, ui ) {} );
```

over(event, ui)Type: `dropover`

此事件会在一个允许的拖动（draggable）元素经过这个放置（droppable）对象时触发（基于 `tolerance` 选项）。

- **event**Type: Event
- **ui**Type: Object
  - **draggable**Type: jQuery一个jQuery对象代表的拖动元素。
  - **helper**Type: jQuery一个jQuery对象代表被拖动元素的助手。
  - **position**Type: Object当前可拖动助手的CSS的position（位置）对象，如 `{ top, left }`。
  - **offset**Type: Object当前可拖动助手的偏移位置对象，如 `{ top, left }`。

Code examples:

使用指定的over 回调初始化一个droppable：

```
$( ".selector" ).droppable({
 over: function( event, ui ) {}
});
```

绑定一个事件监听到dropover事件：

```
$( ".selector" ).on( "dropover", function( event, ui ) {} );
```

Example:

A pair of draggable and droppable elements.

```
<!doctype html>
<html lang="en">
<head>
 <meta charset="utf-8">
 <title>droppable demo</title>
 <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-u
 <style>
 #draggable {
 width: 100px;
 height: 100px;
 background: #ccc;
 }
 #droppable {
 position: absolute;
 left: 250px;
 top: 0;
 width: 125px;
 height: 125px;
 background: #999;
 color: #fff;
 padding: 10px;
 }
 </style>
 <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
 <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<div id="droppable">Drop here</div>
<div id="draggable">Drag me</div>

<script>
$( "#draggable" ).draggable();
$( "#droppable" ).droppable({
 drop: function() {
 alert( "dropped" );
 }
});
</script>

</body>
</html>
```

Mouse Interaction

Categories: Interactions | Utilities

Description: 底层的交互组件

QuickNav

Options

- cancel
- delay
- distance

Methods

- _mouseCapture
- _mouseDelayMet
- _mouseDestroy
- _mouseDistanceMet
- _mouseDown
- _mouseDrag
- _mouseInit
- _mouseMove
- _mouseStart
- _mouseStop
- _mouseUp

Events

像 `jQuery.Widget`, Mouse Interaction（鼠标交互）的目的不是直接使用。这纯粹是一个让其他部件（widget）继承基础组件。这个页面只文档被添加到 `jQuery.Widget`，但它包括不打算被重写的内部方法。对外公开的API 是 `_mouseStart()`, `_mouseDrag()`, `_mouseStop()`, 和 `_mouseCapture()`.

Dependencies

- Widget Factory

Options

cancelType: Selector

Default: `"input,textarea,button,select,option"` 防止在指定的元素上相互作用。**Code examples:**

使用指定的 `cancel` 参数初始化一个:

```
$( ".selector" ).mouse({ cancel: ".title" });
```

在初始化后设置或者获取 `cancel` 参数：

```
// getter
var cancel = $( ".selector" ).mouse( "option", "cancel" );

// setter
$( ".selector" ).mouse( "option", "cancel", ".title" );
```

delayType: Number

Default: `0` 时间（以毫秒为单位），当鼠标按下后直到的互动（interactions）激活。此选项可用来阻止当点击一个元素时可能发生的非期望互动（interactions）行为。**Code examples:**

使用指定的 `delay` 参数初始化一个:

```
$( ".selector" ).mouse({ delay: 300 });
```

在初始化后设置或者获取 `delay` 参数：

```
// getter
var delay = $( ".selector" ).mouse( "option", "delay" );

// setter
$( ".selector" ).mouse( "option", "delay", 300 );
```

distanceType: Number

Default: `1` 当鼠标点下后，只有移动指定像素后才开始激活互动（interactions）动作，单位为像素。此选项可用来阻止当点击一个元素时可能发生的非期望拖动行为。**Code examples:**

使用指定的 `distance` 参数初始化一个:

```
$( ".selector" ).mouse({ distance: 10 });
```

在初始化后设置或者获取 `distance` 参数：

```
// getter
var distance = $( ".selector" ).mouse( "option", "distance" );

// setter
$( ".selector" ).mouse( "option", "distance", 10 );
```

Methods

_mouseCapture()Returns: Boolean

确定一个互动（interaction）是否基于事件目标元素。默认总是返回 `true` 。

- 这个方法不接受任何参数。

Code examples:

调用 _mouseCapture 方法:

```
$( ".selector" ).mouse( "_mouseCapture" );
```

_mouseDelayMet()Returns: Boolean

确定 `delay` 选项是否满足当前的互动（interaction）。

- 这个方法不接受任何参数。

Code examples:

调用 _mouseDelayMet 方法:

```
$( ".selector" ).mouse( "_mouseDelayMet" );
```

_mouseDestroy()

破坏互动（interaction）的事件处理程序。这个必须调用从widget的 `_destroy()` 方法。Destroys the interaction event handlers. This must be called from the extending widget's `_destroy()` method.

- 这个方法不接受任何参数。

Code examples:

调用 _mouseDestroy 方法:

```
$( ".selector" ).mouse( "_mouseDestroy" );
```

_mouseDistanceMet()Returns: Boolean

确认 `distance` 选项是否已满足当前的互动（interaction）

- 这个方法不接受任何参数。

Code examples:

调用 _mouseDistanceMet 方法:

```
$( ".selector" ).mouse( "_mouseDistanceMet" );
```

_mouseDown()

互动（interaction）开始处理。验证该事件原先相关的鼠标按钮键 并确保 `delay` 和 `distance` 选项符合之前激活的互动（interaction）。当互动（interaction）准备开始时，调用 `_mouseStart()` 方法来处理扩展部件。

- 这个方法不接受任何参数。

Code examples:

调用 _mouseDown 方法:

```
$( ".selector" ).mouse( "_mouseDown" );
```

_mouseDrag()

扩展部件需要执行一个 `_mouseDrag()` 方法来处理互动（interaction）的每次移动。此方法将接收鼠标移动相关事件。

- 这个方法不接受任何参数。

Code examples:

调用 _mouseDrag 方法:

```
$( ".selector" ).mouse( "_mouseDrag" );
```

_mouseInit()

初始化的互动（interaction）事件处理程序。这必须调用扩展部件的 `_create()` 方法。

- 这个方法不接受任何参数。

Code examples:

调用 _mouseInit 方法:

```
$( ".selector" ).mouse( "_mouseInit" );
```

_mouseMove()

处理互动（interaction）的每次移动。为扩展部件处理器调用 `mouseDrag()` 方法。

- 这个方法不接受任何参数。

Code examples:

调用 _mouseMove 方法:

```
$( ".selector" ).mouse( "_mouseMove" );
```

_mouseStart()

扩展部件需要执行一个 `_mouseStart()` 方法来处理互动（interaction）的开始激活。此方法将接收互动（interaction）开始激活的相关鼠标事件。

- 这个方法不接受任何参数。

Code examples:

调用 _mouseStart 方法:

```
$( ".selector" ).mouse( "_mouseStart" );
```

_mouseStop()

扩展部件需要执行一个 `_mouseStop()` 方法来处理互动（interaction）的结束。此方法将接收互动（interaction）结束的相关鼠标事件。

- 这个方法不接受任何参数。

Code examples:

调用 _mouseStop 方法:

```
$( ".selector" ).mouse( "_mouseStop" );
```

_mouseUp()

互动（interaction）结束的处理程序。调用 `mouseStop()` 方法来处理扩展部件。

- 这个方法不接受任何参数。

Code examples:

调用 _mouseUp 方法:

```
$( ".selector" ).mouse( "_mouseUp" );
```

Resizable Widget

Categories: Interactions

version added: 1.0

Description: 使用鼠标改变一个元素的尺寸。

QuickNavExamples

Options

- alsoResize
- animate
- animateDuration
- animateEasing
- aspectRatio
- autoHide
- cancel
- containment
- delay
- disabled
- distance
- ghost
- grid
- handles
- helper
- maxHeight
- maxWidth
- minHeight
- minWidth

Methods

- destroy
- disable
- enable
- option

- widget

Events

- create
- resize
- start
- stop

jQuery UI Resizable 插件使选定的内容可以调整大小(这以为着那么拥有一些可以拖动的手柄). 你可以指定一个或者多个操作手柄以及指定最小和最大宽度与高度.

Dependencies

- UI Core
- Widget Factory
- Mouse Interaction

其他注意事项：

- 这个widget需要一些功能性的CSS，否则将无法正常工作。 如果你建立一个自定义的主题，使用widget指定的CSS文件作为一个激活点。

Options

alsoResizeType: Selector or jQuery or Element

Default: `false` 在重置元素尺寸大小的同时重置指定的一个或多个元素的尺寸大小。**Code examples:**

使用指定的 `alsoResize` 参数初始化resizable：

```
$( ".selector" ).resizable({ alsoResize: "#mirror" });
```

在初始化后设置或者获取 `alsoResize` 参数：

```
// getter
var alsoResize = $( ".selector" ).resizable( "option", "alsoResize" );

// setter
$( ".selector" ).resizable( "option", "alsoResize", "#mirror" );
```

animateType: Boolean

Default: `false` 在调整大小后使用一段动画完成调整.**Code examples:**

使用指定的 `animate` 参数初始化resizable：

```
$( ".selector" ).resizable({ animate: true });
```

在初始化后设置或者获取 `animate` 参数：

```
// getter
var animate = $( ".selector" ).resizable( "option", "animate" );

// setter
$( ".selector" ).resizable( "option", "animate", true );
```

animateDurationType: Number or String

Default: `"slow"` 当使用 `animate` 选项时，动画持续的时间。单位毫秒。允许使用的值:

- **Number**: 毫秒数。
- **String**: 一个表示持续时间的字符串，比如 `"slow"` or `"fast"` 。

Code examples:

使用指定的 `animateDuration` 参数初始化resizable：

```
$( ".selector" ).resizable({ animateDuration: "fast" });
```

在初始化后设置或者获取 `animateDuration` 参数：

```
// getter
var animateDuration = $( ".selector" ).resizable( "option", "animateDuration" );

// setter
$( ".selector" ).resizable( "option", "animateDuration", "fast" );
```

animateEasingType: String

Default: `"swing"` 动画执行时的缓冲效果。当使用 `animate` 选项时，哪个easing （缓冲函数）被应用。**Code examples:**

使用指定的 `animateEasing` 参数初始化resizable：

```
$( ".selector" ).resizable({ animateEasing: "easeOutBounce" });
```

在初始化后设置或者获取 `animateEasing` 参数：

```
// getter
var animateEasing = $( ".selector" ).resizable( "option", "animateEasing" );

// setter
$( ".selector" ).resizable( "option", "animateEasing", "easeOutBounce" );
```

aspectRatioType: Boolean or Number

Default: `false` 该元素是否应限制在一个特定的比例进行缩放。允许使用的值:

- **Boolean**: 如果设置为 `true` , 大小将按照原先的宽高比进行调整。
- **Number**: 强制元素调整大小时保持一个特定的宽高比。

Code examples:

使用指定的 `aspectRatio` 参数初始化resizable：

```
$( ".selector" ).resizable({ aspectRatio: true });
```

在初始化后设置或者获取 `aspectRatio` 参数：

```
// getter
var aspectRatio = $( ".selector" ).resizable( "option", "aspectRatio" );

// setter
$( ".selector" ).resizable( "option", "aspectRatio", true );
```

autoHideType: Boolean

Default: `false` 如果设置为真, 将会自动隐藏调整手柄图标,除非鼠标移动到该元素上.**Code examples:**

使用指定的 `autoHide` 参数初始化resizable：

```
$( ".selector" ).resizable({ autoHide: true });
```

在初始化后设置或者获取 `autoHide` 参数：

```
// getter
var autoHide = $( ".selector" ).resizable( "option", "autoHide" );

// setter
$( ".selector" ).resizable( "option", "autoHide", true );
```

cancelType: Selector

Default: `"input,textarea,button,select,option"` 如果设置了选择器匹配,将拒绝对匹配元素的大小调整.**Code examples:**

使用指定的 `cancel` 参数初始化resizable：

```
$( ".selector" ).resizable({ cancel: ".cancel" });
```

在初始化后设置或者获取 `cancel` 参数：

```
// getter
var cancel = $( ".selector" ).resizable( "option", "cancel" );

// setter
$( ".selector" ).resizable( "option", "cancel", ".cancel" );
```

containmentType: Selector or Element or String

Default: `false` 使用指定的元素强制性限制大小调整的界限.允许使用的值：

- **Selector**:resizable元素将被限制在该选择器匹配的第一个元素的边界内。 如果没有匹配的元素，那么设置将不起作用。
- **Element**: resizable元素将被限制在这个元素的边界内。
- **String**: 可能的值： `"parent"` 和 `"document"`.

Code examples:

使用指定的 `containment` 参数初始化resizable：

```
$( ".selector" ).resizable({ containment: "parent" });
```

在初始化后设置或者获取 `containment` 参数：

```
// getter
var containment = $( ".selector" ).resizable( "option", "containment" );

// setter
$( ".selector" ).resizable( "option", "containment", "parent" );
```

delayType: Number

Default: `0` 设定需要经过多少毫秒以后调整才会开始. 如果指定了该参数, 调整不会马上开始, 除非鼠标调整动作已经持续了指定的时间.这可以防止误操作对元素进行了非预期的调整.**Code examples:**

使用指定的 `delay` 参数初始化resizable：

```
$( ".selector" ).resizable({ delay: 150 });
```

在初始化后设置或者获取 `delay` 参数：

```
// getter
var delay = $( ".selector" ).resizable( "option", "delay" );

// setter
$( ".selector" ).resizable( "option", "delay", 150 );
```

disabledType: Boolean

Default: `false` 如果设置为 `true` 将禁止resizable（缩放）。**Code examples:**

使用指定的 `disabled` 参数初始化resizable：

```
$( ".selector" ).resizable({ disabled: true });
```

在初始化后设置或者获取 `disabled` 参数：

```
// getter
var disabled = $( ".selector" ).resizable( "option", "disabled" );

// setter
$( ".selector" ).resizable( "option", "disabled", true );
```

distanceType: Number

Default: `1` 设定调整操作需要移动多少个像素后调整才会开始. 如果指定了该参数, 调整不会马上开始,直到鼠标移动了指定的像素后.这可以防止误操作对元素进行了非预期的调整.**Code examples:**

使用指定的 `distance` 参数初始化resizable：

```
$( ".selector" ).resizable({ distance: 30 });
```

在初始化后设置或者获取 `distance` 参数：

```
// getter
var distance = $( ".selector" ).resizable( "option", "distance" );

// setter
$( ".selector" ).resizable( "option", "distance", 30 );
```

ghostType: Boolean

Default: `false` 如果设置为 `true`，将会在调整过程中看到一个半透明的辅助元素。**Code examples:**

使用指定的 `ghost` 参数初始化resizable：

```
$( ".selector" ).resizable({ ghost: true });
```

在初始化后设置或者获取 `ghost` 参数：

```
// getter
var ghost = $( ".selector" ).resizable( "option", "ghost" );

// setter
$( ".selector" ).resizable( "option", "ghost", true );
```

gridType: Array

Default: `false` 设置调整x和y改变的像素. 调整大小的元素到网格，每x和y个像素。数组值: [x, y]。**Code examples:**

使用指定的 `grid` 参数初始化resizable：

```
$( ".selector" ).resizable({ grid: [ 20, 10 ] });
```

在初始化后设置或者获取 `grid` 参数：

```
// getter
var grid = $( ".selector" ).resizable( "option", "grid" );

// setter
$( ".selector" ).resizable( "option", "grid", [ 20, 10 ] );
```

handlesType: String or Object

Default: `"e, s, se"` 哪个处理程序被用来resizing（缩放大小）。允许使用的值：

- **String**: 如果指定一个字符串，应该是下列清单中的组合:'n, e, s, w, ne, se, sw, nw, all',每项之间使用逗号分隔. 必要的手柄将由插件自动生成.
- **Object**:

  如果指定一个对象，要支持下面的键值: { n, e, s, w, ne, se, sw, nw }. 指定的用户调整手柄的任何值应该是一个jQuery选择器匹配的子元素. 如果该操作柄不是resizable的一个子元素，你可以提供一个有效的DOMElement 或者直接提供一个jQuery对象.

  注意: 当生成您自己的手柄，每个手柄必须有 *`ui-resizable-handle`* 样式类，以及适当的 *`ui-resizable-{direction}`* 样式类，例如 *`ui-resizable-s`* 。

Code examples:

使用指定的 `handles` 参数初始化resizable：

```
$( ".selector" ).resizable({ handles: "n, e, s, w" });
```

在初始化后设置或者获取 `handles` 参数：

```
// getter
var handles = $( ".selector" ).resizable( "option", "handles" );

// setter
$( ".selector" ).resizable( "option", "handles", "n, e, s, w" );
```

helperType: String

Default: `false` 为大小调整时的代理元素指定一个css样式.当调整完成时,这些元素将回到它以前的状态.**Code examples:**

使用指定的 `helper` 参数初始化resizable：

```
$( ".selector" ).resizable({ helper: "resizable-helper" });
```

在初始化后设置或者获取 `helper` 参数：

```
// getter
var helper = $( ".selector" ).resizable( "option", "helper" );

// setter
$( ".selector" ).resizable( "option", "helper", "resizable-helper" );
```

maxHeightType: Number

Default: `null` 为大小调整设定一个最大高度.**Code examples:**

使用指定的 `maxHeight` 参数初始化resizable：

```
$( ".selector" ).resizable({ maxHeight: 300 });
```

在初始化后设置或者获取 `maxHeight` 参数：

```
// getter
var maxHeight = $( ".selector" ).resizable( "option", "maxHeight" );

// setter
$( ".selector" ).resizable( "option", "maxHeight", 300 );
```

maxWidthType: Number

Default: `null` 为大小调整设定一个最大宽度.**Code examples:**

使用指定的 `maxWidth` 参数初始化resizable：

```
$( ".selector" ).resizable({ maxWidth: 300 });
```

在初始化后设置或者获取 `maxWidth` 参数：

```
// getter
var maxWidth = $( ".selector" ).resizable( "option", "maxWidth" );

// setter
$( ".selector" ).resizable( "option", "maxWidth", 300 );
```

minHeightType: Number

Default: `10` 为大小调整设定一个最小高度.**Code examples:**

使用指定的 `minHeight` 参数初始化resizable：

```
$( ".selector" ).resizable({ minHeight: 150 });
```

在初始化后设置或者获取 `minHeight` 参数：

```
// getter
var minHeight = $( ".selector" ).resizable( "option", "minHeight" );

// setter
$( ".selector" ).resizable( "option", "minHeight", 150 );
```

minWidthType: Number

Default: `10` 为大小调整设定一个最小宽度.**Code examples:**

使用指定的 `minWidth` 参数初始化resizable：

```
$( ".selector" ).resizable({ minWidth: 150 });
```

在初始化后设置或者获取 `minWidth` 参数：

```
// getter
var minWidth = $( ".selector" ).resizable( "option", "minWidth" );

// setter
$( ".selector" ).resizable( "option", "minWidth", 150 );
```

Methods

destroy()

完全移除调整功能. 这将使元素返回到之前的初始化状态.

- 这个方法不接受任何参数。

Code examples:

调用 destroy 方法：

```
$( ".selector" ).resizable( "destroy" );
```

disable()

关闭resizable.

- 这个方法不接受任何参数。

Code examples:

调用 disable 方法：

```
$( ".selector" ).resizable( "disable" );
```

enable()

打开resizable.

- 这个方法不接受任何参数。

Code examples:

调用 enable 方法：

```
$( ".selector" ).resizable( "enable" );
```

option(optionName)Returns: Object

通过指定的 `optionName` ，获取当前关联的值。

- **optionName**Type: String要获取值的选项名

Code examples:

调用这个方法：

```
var isDisabled = $( ".selector" ).resizable( "option", "disabled" );
```

option()Returns: PlainObject

获取一个对象，它包含表示当前resizable的选项hash的键/值对。

- 这个方法不接受任何参数。

Code examples:

调用这个方法：

```
var options = $( ".selector" ).resizable( "option" );
```

option(optionName, value)

通过指定的 `optionName`，设置resizable的相关选项值。

- **optionName**Type: String要设置值的选项名。
- **value**Type: Object要设置选项的值。

Code examples:

调用这个方法：

```
$( ".selector" ).resizable( "option", "disabled", true );
```

option(options)

为resizable设置一个或多个选项。

- **options**Type: Object设置的选项/值对的对象。

Code examples:

调用这个方法：

```
$( ".selector" ).resizable( "option", { disabled: true } );
```

widget()Returns: jQuery

返回一个 `jQuery`，它包含了resizable元素。

- 这个方法不接受任何参数。

Code examples:

调用 widget 方法：

```
var widget = $( ".selector" ).resizable( "widget" );
```

Events

create(event, ui)Type: `resizecreate`

此事件会在resizable创建时触发。

- **event**Type: Event
- **ui**Type: Object

注意：`ui` 对象是空对象，包括是为了和其他事件的一致性。

Code examples:

使用指定的 create 回调初始化一个resizable：

```
$( ".selector" ).resizable({
  create: function( event, ui ) {}
});
```

绑定一个事件监听到resizecreate事件：

```
$( ".selector" ).on( "resizecreate", function( event, ui ) {} );
```

resize(event, ui)Type: `resize`

这个事件将在拖动手柄进行调整时触发.

- **event**Type: Event
- **ui**Type: Object
    - **element**Type: jQuery一个jQuery对象代表被 resized 的元素。
    - **helper**Type: jQuery一个jQuery对象代表被resized元素的助手。
    - **originalElement**Type: jQuery一个jQuery对象代表被包裹前原先的元素。
    - **originalPosition**Type: Objectresizable元素被resized（缩放）前的CSS的position（位置）对象，如 `{ left, top }` 。
    - **originalSize**Type: Objectresizable元素被resized（缩放）前的尺寸对象，如 `{ width, height }` 。

  - **position**Type: Object当前可resizable（缩放）元素的CSS的position（位置）对象， 如{ top, left }。
  - **size**Type: Object当前可resizable（缩放）元素的尺寸对象， `{ width, height }` 。

Code examples:

使用指定的 resize 回调初始化一个resizable：

```
$( ".selector" ).resizable({
  resize: function( event, ui ) {}
});
```

绑定一个事件监听到resize事件：

```
$( ".selector" ).on( "resize", function( event, ui ) {} );
```

start(event, ui)Type: `resizestart`

这个事件将在调整操作开始时触发.

- **event**Type: Event
- **ui**Type: Object
  - **element**Type: jQuery一个jQuery对象代表被 resized 的元素。
  - **helper**Type: jQuery一个jQuery对象代表被resized元素的助手。
  - **originalElement**Type: jQuery一个jQuery对象代表被包裹前原先的元素。
  - **originalPosition**Type: Objectresizable元素被resized（缩放）前的CSS的position（位置）对象，如 `{ left, top }` 。
  - **originalSize**Type: Objectresizable元素被resized（缩放）前的尺寸对象，如 `{ width, height }` 。
  - **position**Type: Object当前可resizable（缩放）元素的CSS的position（位置）对象， 如{ top, left }。
  - **size**Type: Object当前可resizable（缩放）元素的尺寸对象， `{ width, height }` 。

Code examples:

使用指定的 start 回调初始化一个resizable：

```
$( ".selector" ).resizable({
  start: function( event, ui ) {}
});
```

绑定一个事件监听到 resizestart 事件：

```
$( ".selector" ).on( "resizestart", function( event, ui ) {} );
```

stop(event, ui)Type: `resizestop`

这个事件将在调整操作结束后触发.

- **event**Type: Event
- **ui**Type: Object
  - **element**Type: jQuery一个jQuery对象代表被 resized 的元素。
  - **helper**Type: jQuery一个jQuery对象代表被resized元素的助手。
  - **originalElement**Type: jQuery一个jQuery对象代表被包裹前原先的元素。
  - **originalPosition**Type: Objectresizable元素被resized（缩放）前的CSS的position（位置）对象，如 `{ left, top }` 。
  - **originalSize**Type: Objectresizable元素被resized（缩放）前的尺寸对象，如 `{ width, height }` 。
  - **position**Type: Object当前可resizable（缩放）元素的CSS的position（位置）对象，如{ top, left }。
  - **size**Type: Object当前可resizable（缩放）元素的尺寸对象， `{ width, height }` 。

Code examples:

使用指定的 stop 回调初始化一个resizable：

```
$( ".selector" ).resizable({
  stop: function( event, ui ) {}
});
```

绑定一个事件监听到 resizestop 事件：

```
$( ".selector" ).on( "resizestop", function( event, ui ) {} );
```

Example:

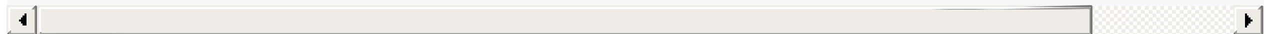
A simple jQuery UI Resizable.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>resizable demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  #resizable {
    width: 100px;
    height: 100px;
    background: #ccc;
}    </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<div id="resizable"></div>

<script>
$( "#resizable" ).resizable();
</script>

</body>
</html>
```



Resizable Widget

Categories: Interactions

version added: 1.0

Description: 使用鼠标改变一个元素的尺寸。

QuickNavExamples

Options

- alsoResize
- animate
- animateDuration
- animateEasing
- aspectRatio
- autoHide
- cancel
- containment
- delay
- disabled
- distance
- ghost
- grid
- handles
- helper
- maxHeight
- maxWidth
- minHeight
- minWidth

Methods

- destroy
- disable
- enable
- option

- widget

Events

- create
- resize
- start
- stop

jQuery UI Resizable 插件使选定的内容可以调整大小(这以为着那么拥有一些可以拖动的手柄). 你可以指定一个或者多个操作手柄以及指定最小和最大宽度与高度.

Dependencies

- UI Core
- Widget Factory
- Mouse Interaction

其他注意事项：

- 这个widget需要一些功能性的CSS，否则将无法正常工作。 如果你建立一个自定义的主题，使用widget指定的CSS文件作为一个激活点。

Options

alsoResizeType: Selector or jQuery or Element

Default: `false` 在重置元素尺寸大小的同时重置指定的一个或多个元素的尺寸大小。**Code examples:**

使用指定的 `alsoResize` 参数初始化resizable：

```
$( ".selector" ).resizable({ alsoResize: "#mirror" });
```

在初始化后设置或者获取 `alsoResize` 参数：

```
// getter
var alsoResize = $( ".selector" ).resizable( "option", "alsoResize" );

// setter
$( ".selector" ).resizable( "option", "alsoResize", "#mirror" );
```

animateType: Boolean

Default: `false` 在调整大小后使用一段动画完成调整.**Code examples:**

使用指定的 `animate` 参数初始化resizable：

```
$( ".selector" ).resizable({ animate: true });
```

在初始化后设置或者获取 `animate` 参数：

```
// getter
var animate = $( ".selector" ).resizable( "option", "animate" );

// setter
$( ".selector" ).resizable( "option", "animate", true );
```

animateDurationType: Number or String

Default: `"slow"` 当使用 `animate` 选项时，动画持续的时间。单位毫秒。允许使用的值:

- **Number**: 毫秒数。
- **String**: 一个表示持续时间的字符串，比如 `"slow"` or `"fast"` 。

Code examples:

使用指定的 `animateDuration` 参数初始化resizable：

```
$( ".selector" ).resizable({ animateDuration: "fast" });
```

在初始化后设置或者获取 `animateDuration` 参数：

```
// getter
var animateDuration = $( ".selector" ).resizable( "option", "animateDuration" );

// setter
$( ".selector" ).resizable( "option", "animateDuration", "fast" );
```

animateEasingType: String

Default: `"swing"` 动画执行时的缓冲效果。当使用 `animate` 选项时，哪个easing （缓冲函数）被应用。**Code examples:**

使用指定的 `animateEasing` 参数初始化resizable：

```
$( ".selector" ).resizable({ animateEasing: "easeOutBounce" });
```

在初始化后设置或者获取 `animateEasing` 参数：

```
// getter
var animateEasing = $( ".selector" ).resizable( "option", "animateEasing" );

// setter
$( ".selector" ).resizable( "option", "animateEasing", "easeOutBounce" );
```

aspectRatioType: Boolean or Number

Default: `false` 该元素是否应限制在一个特定的比例进行缩放。允许使用的值:

- **Boolean**: 如果设置为 `true` , 大小将按照原先的宽高比进行调整。
- **Number**: 强制元素调整大小时保持一个特定的宽高比。

Code examples:

使用指定的 `aspectRatio` 参数初始化resizable：

```
$( ".selector" ).resizable({ aspectRatio: true });
```

在初始化后设置或者获取 `aspectRatio` 参数：

```
// getter
var aspectRatio = $( ".selector" ).resizable( "option", "aspectRatio" );

// setter
$( ".selector" ).resizable( "option", "aspectRatio", true );
```

autoHideType: Boolean

Default: `false` 如果设置为真, 将会自动隐藏调整手柄图标,除非鼠标移动到该元素上.**Code examples:**

使用指定的 `autoHide` 参数初始化resizable：

```
$( ".selector" ).resizable({ autoHide: true });
```

在初始化后设置或者获取 `autoHide` 参数：

```
// getter
var autoHide = $( ".selector" ).resizable( "option", "autoHide" );

// setter
$( ".selector" ).resizable( "option", "autoHide", true );
```

cancelType: Selector

Default: `"input,textarea,button,select,option"` 如果设置了选择器匹配,将拒绝对匹配元素的大小调整.**Code examples:**

使用指定的 `cancel` 参数初始化resizable：

```
$( ".selector" ).resizable({ cancel: ".cancel" });
```

在初始化后设置或者获取 `cancel` 参数：

```
// getter
var cancel = $( ".selector" ).resizable( "option", "cancel" );

// setter
$( ".selector" ).resizable( "option", "cancel", ".cancel" );
```

containmentType: Selector or Element or String

Default: `false` 使用指定的元素强制性限制大小调整的界限.允许使用的值：

- **Selector**:resizable元素将被限制在该选择器匹配的第一个元素的边界内。 如果没有匹配的元素，那么设置将不起作用。
- **Element**: resizable元素将被限制在这个元素的边界内。
- **String**: 可能的值： `"parent"` 和 `"document"`.

Code examples:

使用指定的 `containment` 参数初始化resizable：

```
$( ".selector" ).resizable({ containment: "parent" });
```

在初始化后设置或者获取 `containment` 参数：

```
// getter
var containment = $( ".selector" ).resizable( "option", "containment" );

// setter
$( ".selector" ).resizable( "option", "containment", "parent" );
```

delayType: Number

Default: `0` 设定需要经过多少毫秒以后调整才会开始. 如果指定了该参数, 调整不会马上开始, 除非鼠标调整动作已经持续了指定的时间.这可以防止误操作对元素进行了非预期的调整.**Code examples:**

使用指定的 `delay` 参数初始化resizable：

```
$( ".selector" ).resizable({ delay: 150 });
```

在初始化后设置或者获取 `delay` 参数：

```
// getter
var delay = $( ".selector" ).resizable( "option", "delay" );

// setter
$( ".selector" ).resizable( "option", "delay", 150 );
```

disabledType: Boolean

Default: `false` 如果设置为 `true` 将禁止resizable（缩放）。**Code examples:**

使用指定的 `disabled` 参数初始化resizable：

```
$( ".selector" ).resizable({ disabled: true });
```

在初始化后设置或者获取 `disabled` 参数：

```
// getter
var disabled = $( ".selector" ).resizable( "option", "disabled" );

// setter
$( ".selector" ).resizable( "option", "disabled", true );
```

distanceType: Number

Default: `1` 设定调整操作需要移动多少个像素后调整才会开始. 如果指定了该参数, 调整不会马上开始,直到鼠标移动了指定的像素后.这可以防止误操作对元素进行了非预期的调整.**Code examples:**

使用指定的 `distance` 参数初始化resizable：

```
$( ".selector" ).resizable({ distance: 30 });
```

在初始化后设置或者获取 `distance` 参数：

```
// getter
var distance = $( ".selector" ).resizable( "option", "distance" );

// setter
$( ".selector" ).resizable( "option", "distance", 30 );
```

ghostType: Boolean

Default: `false` 如果设置为 `true`，将会在调整过程中看到一个半透明的辅助元素。**Code examples:**

使用指定的 `ghost` 参数初始化resizable：

```
$( ".selector" ).resizable({ ghost: true });
```

在初始化后设置或者获取 `ghost` 参数：

```
// getter
var ghost = $( ".selector" ).resizable( "option", "ghost" );

// setter
$( ".selector" ).resizable( "option", "ghost", true );
```

gridType: Array

Default: `false` 设置调整x和y改变的像素. 调整大小的元素到网格， 每x和y个像素。数组值: [x, y]。**Code examples:**

使用指定的 `grid` 参数初始化resizable：

```
$( ".selector" ).resizable({ grid: [ 20, 10 ] });
```

在初始化后设置或者获取 `grid` 参数：

```
// getter
var grid = $( ".selector" ).resizable( "option", "grid" );

// setter
$( ".selector" ).resizable( "option", "grid", [ 20, 10 ] );
```

handlesType: String or Object

Default: `"e, s, se"` 哪个处理程序被用来resizing（缩放大小）。允许使用的值：

- **String**: 如果指定一个字符串, 应该是下列清单中的组合:'n, e, s, w, ne, se, sw, nw, all',每项之间使用逗号分隔. 必要的手柄将由插件自动生成.
- **Object**:

  如果指定一个对象, 要支持下面的键值: { n, e, s, w, ne, se, sw, nw }. 指定的用户调整手柄的任何值应该是一个jQuery选择器匹配的子元素. 如果该操作柄不是resizable的一个子元素, 你可以提供一个有效的DOMElement 或者直接提供一个jQuery对象.

  注意: 当生成您自己的手柄，每个手柄必须有 *`ui-resizable-handle`* 样式类，以及适当的 *`ui-resizable-{direction}`* 样式类，例如 *`ui-resizable-s`* 。

Code examples:

使用指定的 `handles` 参数初始化resizable：

```
$( ".selector" ).resizable({ handles: "n, e, s, w" });
```

在初始化后设置或者获取 `handles` 参数：

```
// getter
var handles = $( ".selector" ).resizable( "option", "handles" );

// setter
$( ".selector" ).resizable( "option", "handles", "n, e, s, w" );
```

helperType: String

Default: `false` 为大小调整时的代理元素指定一个css样式.当调整完成时,这些元素将回到它以前的状态.**Code examples:**

使用指定的 `helper` 参数初始化resizable：

```
$( ".selector" ).resizable({ helper: "resizable-helper" });
```

在初始化后设置或者获取 `helper` 参数：

```
// getter
var helper = $( ".selector" ).resizable( "option", "helper" );

// setter
$( ".selector" ).resizable( "option", "helper", "resizable-helper" );
```

maxHeightType: Number

Default: `null` 为大小调整设定一个最大高度.**Code examples:**

使用指定的 `maxHeight` 参数初始化resizable：

```
$( ".selector" ).resizable({ maxHeight: 300 });
```

在初始化后设置或者获取 `maxHeight` 参数：

```
// getter
var maxHeight = $( ".selector" ).resizable( "option", "maxHeight" );

// setter
$( ".selector" ).resizable( "option", "maxHeight", 300 );
```

maxWidthType: Number

Default: `null` 为大小调整设定一个最大宽度.**Code examples:**

使用指定的 `maxWidth` 参数初始化resizable：

```
$( ".selector" ).resizable({ maxWidth: 300 });
```

在初始化后设置或者获取 `maxWidth` 参数：

```
// getter
var maxWidth = $( ".selector" ).resizable( "option", "maxWidth" );

// setter
$( ".selector" ).resizable( "option", "maxWidth", 300 );
```

minHeightType: Number

Default: `10` 为大小调整设定一个最小高度.**Code examples:**

使用指定的 `minHeight` 参数初始化resizable：

```
$( ".selector" ).resizable({ minHeight: 150 });
```

在初始化后设置或者获取 `minHeight` 参数：

```
// getter
var minHeight = $( ".selector" ).resizable( "option", "minHeight" );

// setter
$( ".selector" ).resizable( "option", "minHeight", 150 );
```

minWidthType: Number

Default: `10` 为大小调整设定一个最小宽度.**Code examples:**

使用指定的 `minWidth` 参数初始化resizable：

```
$( ".selector" ).resizable({ minWidth: 150 });
```

在初始化后设置或者获取 `minWidth` 参数：

```
// getter
var minWidth = $( ".selector" ).resizable( "option", "minWidth" );

// setter
$( ".selector" ).resizable( "option", "minWidth", 150 );
```

Methods

destroy()

完全移除调整功能. 这将使元素返回到之前的初始化状态.

- 这个方法不接受任何参数。

Code examples:

调用 destroy 方法：

```
$( ".selector" ).resizable( "destroy" );
```

disable()

关闭resizable.

- 这个方法不接受任何参数。

Code examples:

调用 disable 方法：

```
$( ".selector" ).resizable( "disable" );
```

enable()

打开resizable.

- 这个方法不接受任何参数。

Code examples:

调用 enable 方法：

```
$( ".selector" ).resizable( "enable" );
```

option(optionName)Returns: Object

通过指定的 `optionName` ，获取当前关联的值。

- **optionName**Type: String要获取值的选项名

Code examples:

调用这个方法：

```
var isDisabled = $( ".selector" ).resizable( "option", "disabled" );
```

option()Returns: PlainObject

获取一个对象，它包含表示当前resizable的选项hash的键/值对。

- 这个方法不接受任何参数。

Code examples:

调用这个方法：

```
var options = $( ".selector" ).resizable( "option" );
```

option(optionName, value)

通过指定的 `optionName` ，设置resizable的相关选项值。

- **optionName**Type: String要设置值的选项名。
- **value**Type: Object要设置选项的值。

Code examples:

调用这个方法：

```
$( ".selector" ).resizable( "option", "disabled", true );
```

option(options)

为resizable设置一个或多个选项。

- **options**Type: Object设置的选项/值对的对象。

Code examples:

调用这个方法：

```
$( ".selector" ).resizable( "option", { disabled: true } );
```

widget()Returns: jQuery

返回一个 `jQuery` ，它包含了resizable元素。

- 这个方法不接受任何参数。

Code examples:

调用 widget 方法：

```
var widget = $( ".selector" ).resizable( "widget" );
```

Events

create(event, ui)Type: `resizecreate`

此事件会在resizable创建时触发。

- **event**Type: Event
- **ui**Type: Object

注意：`ui` 对象是空对象，包括是为了和其他事件的一致性。

Code examples:

使用指定的 create 回调初始化一个resizable：

```
$( ".selector" ).resizable({
  create: function( event, ui ) {}
});
```

绑定一个事件监听到resizecreate事件：

```
$( ".selector" ).on( "resizecreate", function( event, ui ) {} );
```

resize(event, ui)Type: `resize`

这个事件将在拖动手柄进行调整时触发.

- **event**Type: Event
- **ui**Type: Object
  - **element**Type: jQuery一个jQuery对象代表被 resized 的元素。
  - **helper**Type: jQuery一个jQuery对象代表被resized元素的助手。
  - **originalElement**Type: jQuery一个jQuery对象代表被包裹前原先的元素。
  - **originalPosition**Type: Objectresizable元素被resized（缩放）前的CSS的position（位置）对象，如 `{ left, top }` 。
  - **originalSize**Type: Objectresizable元素被resized（缩放）前的尺寸对象，如 `{ width, height }` 。

    - **position**Type: Object当前可resizable（缩放）元素的CSS的position（位置）对象，如{ top, left }。
    - **size**Type: Object当前可resizable（缩放）元素的尺寸对象，`{ width, height }`。

Code examples:

使用指定的 resize 回调初始化一个resizable：

```
$( ".selector" ).resizable({
  resize: function( event, ui ) {}
});
```

绑定一个事件监听到resize事件：

```
$( ".selector" ).on( "resize", function( event, ui ) {} );
```

start(event, ui)Type: `resizestart`

这个事件将在调整操作开始时触发.

- **event**Type: Event
- **ui**Type: Object
    - **element**Type: jQuery一个jQuery对象代表被 resized 的元素。
    - **helper**Type: jQuery一个jQuery对象代表被resized元素的助手。
    - **originalElement**Type: jQuery一个jQuery对象代表被包裹前原先的元素。
    - **originalPosition**Type: Objectresizable元素被resized（缩放）前的CSS的position（位置）对象，如`{ left, top }`。
    - **originalSize**Type: Objectresizable元素被resized（缩放）前的尺寸对象，如`{ width, height }`。
    - **position**Type: Object当前可resizable（缩放）元素的CSS的position（位置）对象，如{ top, left }。
    - **size**Type: Object当前可resizable（缩放）元素的尺寸对象，`{ width, height }`。

Code examples:

使用指定的 start 回调初始化一个resizable：

```
$( ".selector" ).resizable({
  start: function( event, ui ) {}
});
```

绑定一个事件监听到 resizestart 事件：

```
$( ".selector" ).on( "resizestart", function( event, ui ) {} );
```

stop(event, ui)Type: `resizestop`

这个事件将在调整操作结束后触发.

- **event**Type: Event
- **ui**Type: Object
    - **element**Type: jQuery一个jQuery对象代表被 resized 的元素。
    - **helper**Type: jQuery一个jQuery对象代表被resized元素的助手。
    - **originalElement**Type: jQuery一个jQuery对象代表被包裹前原先的元素。
    - **originalPosition**Type: Objectresizable元素被resized（缩放）前的CSS的position（位置）对象，如 `{ left, top }` 。
    - **originalSize**Type: Objectresizable元素被resized（缩放）前的尺寸对象，如 `{ width, height }` 。
    - **position**Type: Object当前可resizable（缩放）元素的CSS的position（位置）对象，如{ top, left }。
    - **size**Type: Object当前可resizable（缩放）元素的尺寸对象，`{ width, height }` 。

Code examples:

使用指定的 stop 回调初始化一个resizable：

```
$( ".selector" ).resizable({
  stop: function( event, ui ) {}
});
```

绑定一个事件监听到 resizestop 事件：

```
$( ".selector" ).on( "resizestop", function( event, ui ) {} );
```

Example:

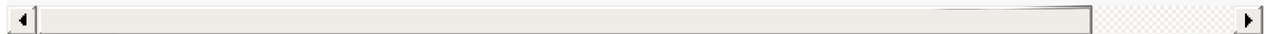
A simple jQuery UI Resizable.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>resizable demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  #resizable {
    width: 100px;
    height: 100px;
    background: #ccc;
}    </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<div id="resizable"></div>

<script>
$( "#resizable" ).resizable();
</script>

</body>
</html>
```



Selectable Widget

Categories: Interactions

version added: 1.0

Description: 使用鼠标选择一个或一组元素。

QuickNavExamples

Options

- appendTo
- autoRefresh
- cancel
- delay
- disabled
- distance
- filter
- tolerance

Methods

- destroy
- disable
- enable
- option
- refresh
- widget

Events

- create
- selected
- selecting
- start
- stop
- unselected

- unselectingjQuery UI Selectable 插件允许一个元素被鼠标划出的选择区域选中. 同样, 元素也可以被点击选中或者同时按住 Ctrl/Meta键, 允许多个(非连续)的选择.

依赖关系

- UI Core
- Widget Factory
- Mouse Interaction

其他注意事项：

- 这个widget需要一些功能性的CSS，否则将无法正常工作。 如果你建立一个自定义的主题，使用widget指定的CSS文件作为一个激活点。

Options

appendToType: Selector

Default: `"body"` 选择帮手（套索） 应追加到哪个元素 。**Code examples:**

使用指定的 `appendTo` 参数初始化selectable：

```
$( ".selector" ).selectable({ appendTo: "#someElem" });
```

在初始化后设置或者获取 `appendTo` 选项：

```
// getter
var appendTo = $( ".selector" ).selectable( "option", "appendTo" );

// setter
$( ".selector" ).selectable( "option", "appendTo", "#someElem" );
```

autoRefreshType: Boolean

Default: `true` 这个选项确定每个选择操作开始时如何刷新(重新计算)每个选择项的位置和大小. 如果你有很多很多选择项, 你应当设置此项为false并且手动调用 `refresh()` 方法.**Code examples:**

使用指定的 `autoRefresh` 参数初始化selectable：

```
$( ".selector" ).selectable({ autoRefresh: false });
```

在初始化后设置或者获取 `autoRefresh` 选项：

```
// getter
var autoRefresh = $( ".selector" ).selectable( "option", "autoRefresh" );

// setter
$( ".selector" ).selectable( "option", "autoRefresh", false );
```

cancelType: Selector

Default: `"input,textarea,button,select,option"` 如果你使用了匹配选择器,符合匹配的元素将被禁止可选.**Code examples:**

使用指定的 `cancel` 参数初始化selectable：

```
$( ".selector" ).selectable({ cancel: "a,.cancel" });
```

在初始化后设置或者获取 `cancel` 选项：

```
// getter
var cancel = $( ".selector" ).selectable( "option", "cancel" );

// setter
$( ".selector" ).selectable( "option", "cancel", "a,.cancel" );
```

delayType: Integer

Default: `0` 定义需要经过多少毫秒后选择才会开始. 这可以预防意外的点击造成元素被选择.**Code examples:**

使用指定的 `delay` 参数初始化selectable：

```
$( ".selector" ).selectable({ delay: 150 });
```

在初始化后设置或者获取 `delay` 选项：

```
// getter
var delay = $( ".selector" ).selectable( "option", "delay" );

// setter
$( ".selector" ).selectable( "option", "delay", 150 );
```

disabledType: Boolean

Default: `false` 如果设置为 `true` 将禁止selectable。**Code examples:**

使用指定的 `disabled` 参数初始化selectable：

```
$( ".selector" ).selectable({ disabled: true });
```

在初始化后设置或者获取 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).selectable( "option", "disabled" );

// setter
$( ".selector" ).selectable( "option", "disabled", true );
```

distanceType: Number

Default: `0` 定义需要移动多少个像素选择才会开始. 如果指定了该项, 选择不会马上开始,而是会在鼠标移动了指定像素的距离之后才会开始.**Code examples:**

使用指定的 `distance` 参数初始化selectable：

```
$( ".selector" ).selectable({ distance: 30 });
```

在初始化后设置或者获取 `distance` 选项：

```
// getter
var distance = $( ".selector" ).selectable( "option", "distance" );

// setter
$( ".selector" ).selectable( "option", "distance", 30 );
```

filterType: Selector

Default: `"*"` 匹配子元素中那些符合条件的元素才可以被选择.**Code examples:**

使用指定的 `filter` 参数初始化selectable：

```
$( ".selector" ).selectable({ filter: "li" });
```

在初始化后设置或者获取 `filter` 选项：

```
// getter
var filter = $( ".selector" ).selectable( "option", "filter" );

// setter
$( ".selector" ).selectable( "option", "filter", "li" );
```

toleranceType: String

Default: `"touch"` 指定那种模式，用来测试套索是否应该选择一个项目。允许使用的值:

- `"fit"`：套索完全重叠的项目。
- `"touch"`：套索重叠的项目任何部分。

Code examples:

使用指定的 `tolerance` 参数初始化selectable：

```
$( ".selector" ).selectable({ tolerance: "fit" });
```

在初始化后设置或者获取 `tolerance` 选项：

```
// getter
var tolerance = $( ".selector" ).selectable( "option", "tolerance" );

// setter
$( ".selector" ).selectable( "option", "tolerance", "fit" );
```

Methods

destroy()

完全移除selectable功能. 这将使元素返回到之前的初始化状态.

- 这个方法不接受任何参数。

Code examples:

调用 destroy 方法:

```
$( ".selector" ).selectable( "destroy" );
```

disable()

禁用selectable.

- 这个方法不接受任何参数。

Code examples:

调用 disable 方法:

```
$( ".selector" ).selectable( "disable" );
```

enable()

启用 selectable.

- 这个方法不接受任何参数。

Code examples:

调用 enable 方法:

```
$( ".selector" ).selectable( "enable" );
```

option(optionName)Returns: Object

通过指定的 `optionName` ，获取当前关联的值。

- **optionName**Type: String要获取值的选项名

Code examples:

调用 方法:

```
var isDisabled = $( ".selector" ).selectable( "option", "disabled" );
```

option()Returns: PlainObject

获取一个对象，它包含表示当前resizable的选项hash的键/值对。

- 这个方法不接受任何参数。

Code examples:

调用这个方法:

```
var options = $( ".selector" ).selectable( "option" );
```

option(optionName, value)

通过指定的 `optionName` ，设置selectable的相关选项值。

- **optionName**Type: String要设置值的选项名。
- **value**Type: Object要设置选项的值。

Code examples:

调用这个方法:

```
$( ".selector" ).selectable( "option", "disabled", true );
```

option(options)

为selectable设置一个或多个选项。

- **options**Type: Object用来设置的选项/值对的对象。

Code examples:

调用这个方法:

```
$( ".selector" ).selectable( "option", { disabled: true } );
```

refresh()

刷新每个选择项的位置和大小. 这个方法用来手动重新计算选择项的位置和大小, 在 `autoRefresh` 设置为 `false` 时很有用。

- 这个方法不接受任何参数。

Code examples:

调用 refresh 方法:

```
$( ".selector" ).selectable( "refresh" );
```

widget()Returns: jQuery

返回一个 `jQuery` ， 它包含了selectable元素。

- 这个方法不接受任何参数。

Code examples:

调用 widget 方法:

```
var widget = $( ".selector" ).selectable( "widget" );
```

Events

create(event, ui)Type: `selectablecreate`

此事件会在 selectable 创建时触发。

- **event**Type: Event

- **ui**Type: Object

注意：`ui` 对象是空对象，包括是为了和其他事件的一致性。

Code examples:

使用指定的 create 回调初始化一个selectable：

```
$( ".selector" ).selectable({
  create: function( event, ui ) {}
});
```

绑定一个事件监听到 selectablecreate 事件：

```
$( ".selector" ).on( "selectablecreate", function( event, ui ) {} );
```

selected(event, ui)Type: `selectableselected`

此事件会在选择操作结束时，在添加到选择的每个元素上触发。

- **event**Type: Event
- **ui**Type: Object
  - **selected**Type: Element已被选择的selectable项。

Code examples:

使用指定的 selected 回调初始化一个selectable：

```
$( ".selector" ).selectable({
  selected: function( event, ui ) {}
});
```

绑定一个事件监听到selectableselected事件：

```
$( ".selector" ).on( "selectableselected", function( event, ui ) {} );
```

selecting(event, ui)Type: `selectableselecting`

此事件会在选择操作过程中，在添加到选择的每个元素上触发。

- **event**Type: Event
- **ui**Type: Object
  - **selecting**Type: Element当前已被选择的selectable项。

Code examples:

使用指定的 selecting 回调初始化一个selectablec：

```
$( ".selector" ).selectable({
  selecting: function( event, ui ) {}
});
```

绑定一个事件监听到 selectableselecting 事件：

```
$( ".selector" ).on( "selectableselecting", function( event, ui ) {} );
```

start(event, ui)Type: `selectablestart`

这个事件将在选择操作开始时触发。

- **event**Type: Event
- **ui**Type: Object

注意：`ui` 对象是空对象，包括是为了和其他事件的一致性。

Code examples:

使用指定的 start 回调初始化一个selectable：

```
$( ".selector" ).selectable({
  start: function( event, ui ) {}
});
```

绑定一个事件监听到 selectablestart 事件：

```
$( ".selector" ).on( "selectablestart", function( event, ui ) {} );
```

stop(event, ui)Type: `selectablestop`

这个事件将在选择操作结束后触发。

- **event**Type: Event
- **ui**Type: Object

注意：`ui` 对象是空对象，包括是为了和其他事件的一致性。

Code examples:

使用指定的 stop 回调初始化一个selectable：

```
$( ".selector" ).selectable({
  stop: function( event, ui ) {}
});
```

绑定一个事件监听到 selectablestop 事件：

```
$( ".selector" ).on( "selectablestop", function( event, ui ) {} );
```

unselected(event, ui)Type: `selectableunselected`

此事件会在选择操作结束时，在从选择元素集合中移除的每个元素上触发。

- **event**Type: Event
- **ui**Type: Object
  - **unselected**Type: Element已被取消选中的可选择项。

Code examples:

使用指定的 unselected 回调初始化一个selectable：

```
$( ".selector" ).selectable({
  unselected: function( event, ui ) {}
});
```

绑定一个事件监听到 selectableunselected 事件：

```
$( ".selector" ).on( "selectableunselected", function( event, ui ) {} );
```

unselecting(event, ui)Type: `selectableunselecting`

此事件会在选择操作过程中，在从选择元素集合中移除的每个元素上触发。

- **event**Type: Event
- **ui**Type: Object
  - **unselecting**Type: Element当前已被取消选中的可选择项。

Code examples:

使用指定的 unselecting 回调初始化一个selectable：

```
$( ".selector" ).selectable({
  unselecting: function( event, ui ) {}
});
```

绑定一个事件监听到 selectableunselecting 事件：

```
$( ".selector" ).on( "selectableunselecting", function( event, ui ) {} );
```

Example:

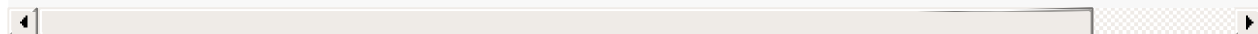
A simple jQuery UI Selectable.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>selectable demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <style>
  #selectable .ui-selecting {
    background: #ccc;
  }
  #selectable .ui-selected {
    background: #999;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<ul id="selectable">
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
  <li>Item 5</li>
</ul>

<script>
$( "#selectable" ).selectable();
</script>

</body>
</html>
```



Sortable Widget

Categories: Interactions

version added: 1.0

Description: 使用鼠标调整列表中或者网格中元素的排序。

QuickNavExamples

Options

- appendTo
- axis
- cancel
- connectWith
- containment
- cursor
- cursorAt
- delay
- disabled
- distance
- dropOnEmpty
- forceHelperSize
- forcePlaceholderSize
- grid
- handle
- helper
- items
- opacity
- placeholder
- revert
- scroll
- scrollSensitivity
- scrollSpeed
- tolerance
- zIndex

Methods

- cancel
- destroy
- disable
- enable
- option
- refresh
- refreshPositions
- serialize
- toArray
- widget

Events

- activate
- beforeStop
- change
- create
- deactivate
- out
- over
- receive
- remove
- sort
- start
- stop
- update

jQuery UI 可排序（Sortable）插件让被被选择的元素通过鼠标拖拽进行排序。

注意：为了排序表格中的行， `tbody` 元素必须作为*sortable*（可排序元素）， 而不是在 `table` 。

Dependencies

- UI Core
- Widget Factory
- Mouse Interaction

Options

appendToType: jQuery or Element or Selector or String

Default: `"parent"` 确定可移动的辅助元素在拖动时可以被添加到何处 (例如, 解决重叠/zIndex问题)。支持多种类型:

- **jQuery**: 一个 jQuery 对象，包含辅助（helper）元素要追加到的元素。
- **Element**: 要被追加辅助（helper）元素的元素。
- **Selector**: 一个选择器，指定哪个元素要追加辅助（helper）元素。
- **String**:字符串 `"parent"` 将促使助手（helper）成为 sortable 项目的同级。

Code examples:

使用指定的 `appendTo` 参数初始化 sortable :

```
$( ".selector" ).sortable({ appendTo: document.body });
```

在初始化后设置或者获取 `appendTo` 参数 :

```
// getter
var appendTo = $( ".selector" ).sortable( "option", "appendTo" );

// setter
$( ".selector" ).sortable( "option", "appendTo", document.body );
```

axisType: String

Default: `false` 如果定义了该参数, 元素可以在水平或垂直方向上实现拖动. 允许使用的值: `"x"` , `"y"` 。**Code examples:**

使用指定的 `axis` 参数初始化 sortable :

```
$( ".selector" ).sortable({ axis: "x" });
```

在初始化后设置或者获取 `axis` 参数 :

```
// getter
var axis = $( ".selector" ).sortable( "option", "axis" );

// setter
$( ".selector" ).sortable( "option", "axis", "x" );
```

cancelType: Selector

Default: `"input,textarea,button,select,option"` 对符合选择器匹配规则的元素不进行排序。**Code examples:**

使用指定的 `cancel` 参数初始化 sortable：

```
$( ".selector" ).sortable({ cancel: "a,button" });
```

在初始化后设置或者获取 `cancel` 参数：

```
// getter
var cancel = $( ".selector" ).sortable( "option", "cancel" );

// setter
$( ".selector" ).sortable( "option", "cancel", "a,button" );
```

connectWithType: Selector

Default: `false` 列表中的项目需被连接的另一个 sortable 元素的选择器。这是一个单向关系，如果您想要项目被双向连接，必须在两个 sortable 元素上都设置 `connectWith` 选项。**Code examples:**

使用指定的 `connectWith` 参数初始化 sortable：

```
$( ".selector" ).sortable({ connectWith: "#shopping-cart" });
```

在初始化后设置或者获取 `connectWith` 参数：

```
// getter
var connectWith = $( ".selector" ).sortable( "option", "connectWith" );

// setter
$( ".selector" ).sortable( "option", "connectWith", "#shopping-cart" );
```

containmentType: Element or Selector or String

Default: `false`

定义一个边界，限制拖动范围在指定的DOM元素内。

注意：为限制拖动范围，指定的元素必须有一个可计算的宽度和高度（但不一定是显式的）。例如，如果你的sortable元素的子元素有 `float: left` 样式，并且指定 `containment: "parent"` ，那么sortable/parent容器必须要有 `float: left` 样式，或者他将有 `height: 0` 样式，导致不确定的行为。

支持多种类型:

- **Element**: 一个用来作为容器的元素。
- **Selector**:一个选择器指定的元素，这个元素用来作为容器
- **String**: 一个字符串指定的元素，这个元素用来作为容器。可能的值: `"parent"` ,

`"document"`, `"window"`。

Code examples:

使用指定的 `containment` 参数初始化 sortable：

```
$( ".selector" ).sortable({ containment: "parent" });
```

在初始化后设置或者获取 `containment` 参数：

```
// getter
var containment = $( ".selector" ).sortable( "option", "containment" );

// setter
$( ".selector" ).sortable( "option", "containment", "parent" );
```

cursorType: String

Default: `"auto"` 定义排序拖动时的鼠标指针样式.**Code examples:**

使用指定的 `cursor` 参数初始化 sortable：

```
$( ".selector" ).sortable({ cursor: "move" });
```

在初始化后设置或者获取 `cursor` 参数：

```
// getter
var cursor = $( ".selector" ).sortable( "option", "cursor" );

// setter
$( ".selector" ).sortable( "option", "cursor", "move" );
```

cursorAtType: Object

Default: `false` 移动排序元素或助手（helper），这样光标总是出现，以便从相同的位置进行拖拽。坐标可通过一个或两个键的组合成一个哈希给出：`{ top, left, right, bottom }`。**Code examples:**

使用指定的 `cursorAt` 参数初始化 sortable：

```
$( ".selector" ).sortable({ cursorAt: { left: 5 } });
```

在初始化后设置或者获取 `cursorAt` 参数：

```
// getter
var cursorAt = $( ".selector" ).sortable( "option", "cursorAt" );

// setter
$( ".selector" ).sortable( "option", "cursorAt", { left: 5 } );
```

delayType: Integer

Default: `0` 在排序拖动开始多少毫秒后元素才开始移动. 这可以防止意外的点击造成元素的拖动.**Code examples:**

使用指定的 `delay` 参数初始化 sortable：

```
$( ".selector" ).sortable({ delay: 150 });
```

在初始化后设置或者获取 `delay` 参数：

```
// getter
var delay = $( ".selector" ).sortable( "option", "delay" );

// setter
$( ".selector" ).sortable( "option", "delay", 150 );
```

disabledType: Boolean

Default: `false` 如果设置为 `true`， 将禁用sortable。**Code examples:**

使用指定的 `disabled` 参数初始化 sortable：

```
$( ".selector" ).sortable({ disabled: true });
```

在初始化后设置或者获取 `disabled` 参数：

```
// getter
var disabled = $( ".selector" ).sortable( "option", "disabled" );

// setter
$( ".selector" ).sortable( "option", "disabled", true );
```

distanceType: Number

Default: `1` 设置当排序拖动开始多少个像素之后元素才开始移动.以像素计。 如果指定了该参数, 排序不会马上开始,直到鼠标移动达到了指定的像素值.**Code examples:**

使用指定的 `distance` 参数初始化 sortable：

```
$( ".selector" ).sortable({ distance: 5 });
```

在初始化后设置或者获取 `distance` 参数：

```
// getter
var distance = $( ".selector" ).sortable( "option", "distance" );

// setter
$( ".selector" ).sortable( "option", "distance", 5 );
```

dropOnEmptyType: Boolean

Default: `true` 如果为 `false`， 这个sortable中项不能拖动到一个空的sortable中。(查看 `connectWith` 选项.**Code examples:**

使用指定的 `dropOnEmpty` 参数初始化 sortable：

```
$( ".selector" ).sortable({ dropOnEmpty: false });
```

在初始化后设置或者获取 `dropOnEmpty` 参数：

```
// getter
var dropOnEmpty = $( ".selector" ).sortable( "option", "dropOnEmpty" );

// setter
$( ".selector" ).sortable( "option", "dropOnEmpty", false );
```

forceHelperSizeType: Boolean

Default: `false` 如果为 `true` , 强迫辅助元素（helper）有一个尺寸大小。**Code examples:**

使用指定的 `forceHelperSize` 参数初始化 sortable：

```
$( ".selector" ).sortable({ forceHelperSize: true });
```

在初始化后设置或者获取 `forceHelperSize` 参数：

```
// getter
var forceHelperSize = $( ".selector" ).sortable( "option", "forceHelperSize" );

// setter
$( ".selector" ).sortable( "option", "forceHelperSize", true );
```

forcePlaceholderSizeType: Boolean

Default: `false` 如果为 `true`，强迫占位符（placeholder）有一个尺寸大小。**Code examples:**

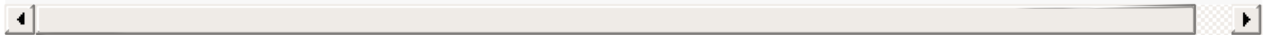
使用指定的 `forcePlaceholderSize` 参数初始化 sortable：

```
$( ".selector" ).sortable({ forcePlaceholderSize: true });
```

在初始化后设置或者获取 `forcePlaceholderSize` 参数：

```
// getter
var forcePlaceholderSize = $( ".selector" ).sortable( "option", "forcePlaceholderSize" );

// setter
$( ".selector" ).sortable( "option", "forcePlaceholderSize", true );
```



gridType: Array

Default: `false` 设置排序对象或者辅助对象（helper）有一个x和y边距,(单位:像素). 数组值: `[ x, y ]`。**Code examples:**

使用指定的 `grid` 参数初始化 sortable：

```
$( ".selector" ).sortable({ grid: [ 20, 10 ] });
```

在初始化后设置或者获取 `grid` 参数：

```
// getter
var grid = $( ".selector" ).sortable( "option", "grid" );

// setter
$( ".selector" ).sortable( "option", "grid", [ 20, 10 ] );
```

handleType: Selector or Element

Default: `false` 如果设定了此参数,那么拖动会在对象内指定的元素上开始.**Code examples:**

使用指定的 `handle` 参数初始化 sortable：

```
$( ".selector" ).sortable({ handle: ".handle" });
```

在初始化后设置或者获取 `handle` 参数：

```
// getter
var handle = $( ".selector" ).sortable( "option", "handle" );

// setter
$( ".selector" ).sortable( "option", "handle", ".handle" );
```

helperType: String or Function()

Default: `"original"` 允许使用一个辅助元素来进行拖动时展示. 所提供的函数在拖动时接受事件和对象元素, 并且需要返回一个DOMElement对象用来当作辅助对象.允许使用的值：

- **String**:如果设置为 `"clone"`，那么这个元素将被克隆，并且克隆出来的元素将被拖动。
- **Function**: 一个函数，将返回拖拽时要使用的 DOMElement。函数接收事件，且元素正被排序。

Code examples:

使用指定的 `helper` 参数初始化 sortable：

```
$( ".selector" ).sortable({ helper: "clone" });
```

在初始化后设置或者获取 `helper` 参数：

```
// getter
var helper = $( ".selector" ).sortable( "option", "helper" );

// setter
$( ".selector" ).sortable( "option", "helper", "clone" );
```

itemsType: Selector

Default: `"> *"` 指定元素内的哪一个项目应是 sortable。**Code examples:**

使用指定的 `items` 参数初始化 sortable：

```
$( ".selector" ).sortable({ items: "> li" });
```

在初始化后设置或者获取 `items` 参数：

```
// getter
var items = $( ".selector" ).sortable( "option", "items" );

// setter
$( ".selector" ).sortable( "option", "items", "> li" );
```

opacityType: Number

Default: `false` 当排序时助手（helper）的不透明度。从 `0.01` 到 `1` 。**Code examples:**

使用指定的 `opacity` 参数初始化 sortable：

```
$( ".selector" ).sortable({ opacity: 0.5 });
```

在初始化后设置或者获取 `opacity` 参数：

```
// getter
var opacity = $( ".selector" ).sortable( "option", "opacity" );

// setter
$( ".selector" ).sortable( "option", "opacity", 0.5 );
```

placeholderType: String

Default: `false` 要应用的 class 名称，否则为白色空白。**Code examples:**

使用指定的 `placeholder` 参数初始化 sortable：

```
$( ".selector" ).sortable({ placeholder: "sortable-placeholder" });
```

在初始化后设置或者获取 `placeholder` 参数：

```
// getter
var placeholder = $( ".selector" ).sortable( "option", "placeholder" );

// setter
$( ".selector" ).sortable( "option", "placeholder", "sortable-placeholder" );
```

revertType: Boolean or Number

Default: `false` sortable 项目是否使用一个流畅的动画还原到它的新位置。支持多个类型：

- **Boolean**:当设置为 `true` ，该项目将会使用动画，动画使用默认的持续时间。
- **Number**: 动画的持续时间，以毫秒计。

Code examples:

使用指定的 `revert` 参数初始化 sortable：

```
$( ".selector" ).sortable({ revert: true });
```

在初始化后设置或者获取 `revert` 参数：

```
// getter
var revert = $( ".selector" ).sortable( "option", "revert" );

// setter
$( ".selector" ).sortable( "option", "revert", true );
```

scrollType: Boolean

Default: `true` 如果设置为 `true`，当到达边缘时页面会滚动。**Code examples:**

使用指定的 `scroll` 参数初始化 sortable：

```
$( ".selector" ).sortable({ scroll: false });
```

在初始化后设置或者获取 `scroll` 参数：

```
// getter
var scroll = $( ".selector" ).sortable( "option", "scroll" );

// setter
$( ".selector" ).sortable( "option", "scroll", false );
```

scrollSensitivityType: Number

Default: `20` 定义鼠标距离边缘多少距离时开始滚动。**Code examples:**

使用指定的 `scrollSensitivity` 参数初始化 sortable：

```
$( ".selector" ).sortable({ scrollSensitivity: 10 });
```

在初始化后设置或者获取 `scrollSensitivity` 参数：

```
// getter
var scrollSensitivity = $( ".selector" ).sortable( "option", "scrollSensitivity" );

// setter
$( ".selector" ).sortable( "option", "scrollSensitivity", 10 );
```

scrollSpeedType: Number

Default: `20` 当鼠标指针获取到在 `scrollSensitivity` 距离内时，窗体滚动的速度。如果 scroll 选项是 false 则忽略。**Code examples:**

使用指定的 `scrollSpeed` 参数初始化 sortable：

```
$( ".selector" ).sortable({ scrollSpeed: 40 });
```

在初始化后设置或者获取 `scrollSpeed` 参数：

```
// getter
var scrollSpeed = $( ".selector" ).sortable( "option", "scrollSpeed" );

// setter
$( ".selector" ).sortable( "option", "scrollSpeed", 40 );
```

toleranceType: String

Default: `"intersect"` 指定用于测试项目被移动时是否覆盖在另一个项目上的模式。可能的值：

- `"intersect"`：项目至少 50% 重叠在其他项目上。
- `"pointer"`：鼠标指针重叠在其他项目上。

Code examples:

使用指定的 `tolerance` 参数初始化 sortable：

```
$( ".selector" ).sortable({ tolerance: "pointer" });
```

在初始化后设置或者获取 `tolerance` 参数：

```
// getter
var tolerance = $( ".selector" ).sortable( "option", "tolerance" );

// setter
$( ".selector" ).sortable( "option", "tolerance", "pointer" );
```

zIndexType: Integer

Default: `1000` 当被排序时，元素/助手（helper）元素的 Z-index。**Code examples:**

使用指定的 `zIndex` 参数初始化 sortable：

```
$( ".selector" ).sortable({ zIndex: 9999 });
```

在初始化后设置或者获取 `zIndex` 参数：

```
// getter
var zIndex = $( ".selector" ).sortable( "option", "zIndex" );

// setter
$( ".selector" ).sortable( "option", "zIndex", 9999 );
```

Methods

cancel()

当前排序开始时，取消一个在当前 sortable 中的改变，且恢复到之前的状态。在 stop 和 receive 回调函数中非常有用。

- 该方法不接受任何参数。

Code examples:

调用 cancel 方法：

```
$( ".selector" ).sortable( "cancel" );
```

destroy()

完全移除 sortable 功能。这会把元素返回到它的预初始化状态。

- 该方法不接受任何参数。

Code examples:

调用 destroy 方法：

```
$( ".selector" ).sortable( "destroy" );
```

disable()

禁用 sortable。

- 该方法不接受任何参数。

Code examples:

调用 disable 方法：

```
$( ".selector" ).sortable( "disable" );
```

enable()

启用 sortable。

- 该方法不接受任何参数。

Code examples:

调用 enable 方法：

```
$( ".selector" ).sortable( "enable" );
```

option(optionName)Returns: Object

获取当前与指定的 `optionName` 关联的值。

- **optionName**Type: String要获取值的选项的名称。

Code examples:

调用该方法：

```
var isDisabled = $( ".selector" ).sortable( "option", "disabled" );
```

option()Returns: PlainObject

获取一个包含键/值对的对象，键/值对表示当前 sortable 选项哈希。

- 该方法不接受任何参数。

Code examples:

调用该方法：

```
var options = $( ".selector" ).sortable( "option" );
```

option(optionName, value)

设置与指定的 `optionName` 关联的 sortable 选项的值。

- **optionName**Type: String要设置的选项的名称。
- **value**Type: Object要为选项设置的值。

Code examples:

调用该方法：

```
$( ".selector" ).sortable( "option", "disabled", true );
```

option(options)

为 sortable 设置一个或多个选项。

- **options**Type: Object要设置的 option-value 对。

Code examples:

调用该方法：

```
$( ".selector" ).sortable( "option", { disabled: true } );
```

refresh()

刷新 sortable 项目。触发所有 sortable 项目重新加载，导致新的项目被认可。

- 该方法不接受任何参数。

Code examples:

调用 refresh 方法：

```
$( ".selector" ).sortable( "refresh" );
```

refreshPositions()

刷新 sortable 项目的缓存位置。调用该方法刷新所有 sortable 的已缓存的项目位置。

- 该方法不接受任何参数。

Code examples:

调用 refreshPositions 方法：

```
$( ".selector" ).sortable( "refreshPositions" );
```

serialize(options)Returns: String

序列化 sortable 的项目 `id` 为一个 form/ajax 可提交的字符串。调用该方法会产生一个可被追加到任何 url 中的哈希，以便简单地把一个新的项目顺序提交回服务器。

默认情况下，它通过每个项目的 `id` 进行工作，id 格式为 `"setname_number"`，且它会产生一个形如 `"setname[]=number&setname[]=number"` 的哈希。

注释：如果序列化返回一个空的字符串，请确认 `id` 属性包含一个下划线（）。形式必须是 *`"set_number"`*。例如，一个 *`id`* 属性为 *`"foo_1"`*、*`"foo_5"`*、*`"foo_2"`* 的 *3* 元素列表将序列化为 *`"foo[]=1&foo[]=5&foo[]=2"`*。您可以使用下划线（）、等号（=）或连字符（-）来分隔集合和数字。例如，`"foo=1"`、`"foo-1"`、`"foo_1"` 所有都序列化为 `"foo[]=1"`。

- **options**Type: Object要自定义序列化的选项。
  - **key**（默认值：`属性中分隔符前面的部分`） 类型：String 描述：把 `part1[]` 替换为指定

的值。
  - **attribute**（默认值：`"id"`） 类型：String 描述：值要使用的属性名称。
  - **expression**（默认值：`/(.+)[-=_](.+)/`） 类型：RegExp 描述：用于把属性值分隔为键和值两部分的正则表达式。

Code examples:

调用 serialize 方法：

```
var sorted = $( ".selector" ).sortable( "serialize", { key: "sort" } );
```

toArray(options)Returns: Array

序列化 sortable 的项目 id 为一个字符串的数组。

- **options**Type: Object要自定义序列化的选项。
  - **attribute** (default: `"id"`)Type: String值要使用的属性名称。

Code examples:

调用 toArray 方法：

```
var sortedIDs = $( ".selector" ).sortable( "toArray" );
```

widget()Returns: jQuery

返回一个包含 sortable 元素的 `jQuery` 对象。

- 该方法不接受任何参数。

Code examples:

调用 widget 方法：

```
var widget = $( ".selector" ).sortable( "widget" );
```

Events

activate(event, ui)Type: `sortactivate`

当使用被连接列表时触发该事件，每个被连接列表在拖拽开始时接收它。

- **event**Type: Event
- **ui**Type: Object

  - **helper**Type: jQueryjQuery 对象，表示被排序的助手（helper）。
  - **item**Type: jQueryjQuery 对象，表示当前被拖拽的元素。
  - **offset**Type: Object助手（helper）的当前的绝对位置，表示为 `{ top, left }`。
  - **position**Type: Object助手（helper）的当前位置，表示为 `{ top, left }`.
  - **originalPosition**Type: Object元素的原始位置，表示为 `{ top, left }`.
  - **sender**Type: jQuery如果从一个 sortable 移动到另一个 sortable，项目来自的那个
  - **placeholder**Type: jQueryjQuery 对象，表示被作为占位符使用的元素。

Code examples:

使用指定的 activate 回调的 sortable：

```
$( ".selector" ).sortable({
  activate: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortactivate 事件：

```
$( ".selector" ).on( "sortactivate", function( event, ui ) {} );
```

beforeStop(event, ui)Type: `sortbeforestop`

当排序停止时触发该事件，除了当占位符（placeholder）/助手（helper）仍然可用时。

- **event**Type: Event
- **ui**Type: Object
  - **helper**Type: jQueryjQuery 对象，表示被排序的助手（helper）。
  - **item**Type: jQueryjQuery 对象，表示当前被拖拽的元素。
  - **offset**Type: Object助手（helper）的当前的绝对位置，表示为 `{ top, left }`。
  - **position**Type: Object助手（helper）的当前位置，表示为 `{ top, left }`.
  - **originalPosition**Type: Object元素的原始位置，表示为 `{ top, left }`.
  - **sender**Type: jQuery如果从一个 sortable 移动到另一个 sortable，项目来自的那个
  - **placeholder**Type: jQueryjQuery 对象，表示被作为占位符使用的元素。

Code examples:

使用指定的 beforeStop 回调的 sortable：

```
$( ".selector" ).sortable({
  beforeStop: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortbeforestop 事件：

```
$( ".selector" ).on( "sortbeforestop", function( event, ui ) {} );
```

change(event, ui)Type: `sortchange`

在排序期间触发该事件，除了当 DOM 位置改变时。

- **event**Type: Event
- **ui**Type: Object
    - **helper**Type: jQueryjQuery 对象，表示被排序的助手（helper）。
    - **item**Type: jQueryjQuery 对象，表示当前被拖拽的元素。
    - **offset**Type: Object助手（helper）的当前的绝对位置，表示为 `{ top, left }`。
    - **position**Type: Object助手（helper）的当前位置，表示为 `{ top, left }`.
    - **originalPosition**Type: Object元素的原始位置，表示为 `{ top, left }`.
    - **sender**Type: jQuery如果从一个 sortable 移动到另一个 sortable，项目来自的那个
    - **placeholder**Type: jQueryjQuery 对象，表示被作为占位符使用的元素。

Code examples:

使用指定的 change 回调的 sortable：

```
$( ".selector" ).sortable({
  change: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortchange 事件：

```
$( ".selector" ).on( "sortchange", function( event, ui ) {} );
```

create(event, ui)Type: `sortcreate`

当 droppable 被创建时触发。

- **event**Type: Event
- **ui**Type: Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

Code examples:

使用指定的 create 回调的 sortable：

```
$( ".selector" ).sortable({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortcreate 事件：

```
$( ".selector" ).on( "sortcreate", function( event, ui ) {} );
```

deactivate(event, ui)Type: `sortdeactivate`

当排序停止时触发该事件，该事件传播到所有可能的连接列表。

- **event**Type: Event
- **ui**Type: Object
  - **helper**Type: jQueryjQuery 对象，表示被排序的助手（helper）。
  - **item**Type: jQueryjQuery 对象，表示当前被拖拽的元素。
  - **offset**Type: Object助手（helper）的当前的绝对位置，表示为 `{ top, left }` 。
  - **position**Type: Object助手（helper）的当前位置，表示为 `{ top, left }` .
  - **originalPosition**Type: Object元素的原始位置，表示为 `{ top, left }` .
  - **sender**Type: jQuery如果从一个 sortable 移动到另一个 sortable，项目来自的那个
  - **placeholder**Type: jQueryjQuery 对象，表示被作为占位符使用的元素。

Code examples:

使用指定的 deactivate 回调的 sortable：

```
$( ".selector" ).sortable({
  deactivate: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortdeactivate 事件：

```
$( ".selector" ).on( "sortdeactivate", function( event, ui ) {} );
```

out(event, ui)Type: `sortout`

当一个 sortable 项目从一个 sortable 列表移除时触发该事件。

Note: 当一个 *sortable* 项目被撤销时也会触发该事件。

- **event**Type: Event
- **ui**Type: Object
  - **helper**Type: jQueryjQuery 对象，表示被排序的助手（helper）。
  - **item**Type: jQueryjQuery 对象，表示当前被拖拽的元素。
  - **offset**Type: Object助手（helper）的当前的绝对位置，表示为 `{ top, left }` 。
  - **position**Type: Object助手（helper）的当前位置，表示为 `{ top, left }` .
  - **originalPosition**Type: Object元素的原始位置，表示为 `{ top, left }` .

    - **sender**Type: jQuery如果从一个 sortable 移动到另一个 sortable，项目来自的那个
    - **placeholder**Type: jQueryjQuery 对象，表示被作为占位符使用的元素。

Code examples:

使用指定的 out 回调的 sortable：

```
$( ".selector" ).sortable({
  out: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortout 事件：

```
$( ".selector" ).on( "sortout", function( event, ui ) {} );
```

over(event, ui)Type: `sortover`

当一个 sortable 项目移动到一个 sortable 列表时触发该事件。

- **event**Type: Event
- **ui**Type: Object
    - **helper**Type: jQueryjQuery 对象，表示被排序的助手（helper）。
    - **item**Type: jQueryjQuery 对象，表示当前被拖拽的元素。
    - **offset**Type: Object助手（helper）的当前的绝对位置，表示为 `{ top, left }`。
    - **position**Type: Object助手（helper）的当前位置，表示为 `{ top, left }`.
    - **originalPosition**Type: Object元素的原始位置，表示为 `{ top, left }`.
    - **sender**Type: jQuery如果从一个 sortable 移动到另一个 sortable，项目来自的那个
    - **placeholder**Type: jQueryjQuery 对象，表示被作为占位符使用的元素。

Code examples:

使用指定的 over 回调的 sortable：

```
$( ".selector" ).sortable({
  over: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortover 事件：

```
$( ".selector" ).on( "sortover", function( event, ui ) {} );
```

receive(event, ui)Type: `sortreceive`

当来自一个连接的 sortable 列表的一个项目被放置到另一个列表时触发该事件。后者是事件目标。

- **event**Type: Event
- **ui**Type: Object
    - **helper**Type: jQueryjQuery 对象，表示被排序的助手（helper）。
    - **item**Type: jQueryjQuery 对象，表示当前被拖拽的元素。
    - **offset**Type: Object助手（helper）的当前的绝对位置，表示为 `{ top, left }`。
    - **position**Type: Object助手（helper）的当前位置，表示为 `{ top, left }`.
    - **originalPosition**Type: Object元素的原始位置，表示为 `{ top, left }`.
    - **sender**Type: jQuery如果从一个 sortable 移动到另一个 sortable，项目来自的那个
    - **placeholder**Type: jQueryjQuery 对象，表示被作为占位符使用的元素。

Code examples:

使用指定的 receive 回调的 sortable：

```
$( ".selector" ).sortable({
  receive: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortreceive 事件：

```
$( ".selector" ).on( "sortreceive", function( event, ui ) {} );
```

remove(event, ui)Type: `sortremove`

当来自一个连接的 sortable 列表的一个项目被放置到另一个列表时触发该事件。前者是事件目标。

- **event**Type: Event
- **ui**Type: Object
    - **helper**Type: jQueryjQuery 对象，表示被排序的助手（helper）。
    - **item**Type: jQueryjQuery 对象，表示当前被拖拽的元素。
    - **offset**Type: Object助手（helper）的当前的绝对位置，表示为 `{ top, left }`。
    - **position**Type: Object助手（helper）的当前位置，表示为 `{ top, left }`.
    - **originalPosition**Type: Object元素的原始位置，表示为 `{ top, left }`.
    - **sender**Type: jQuery如果从一个 sortable 移动到另一个 sortable，项目来自的那个
    - **placeholder**Type: jQueryjQuery 对象，表示被作为占位符使用的元素。

Code examples:

使用指定的 remove 回调的 sortable：

```
$( ".selector" ).sortable({
  remove: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortremove 事件：

```
$( ".selector" ).on( "sortremove", function( event, ui ) {} );
```

sort(event, ui)Type: `sort`

在排序期间触发该事件。

- **event**Type: Event
- **ui**Type: Object
  - **helper**Type: jQueryjQuery 对象，表示被排序的助手（helper）。
  - **item**Type: jQueryjQuery 对象，表示当前被拖拽的元素。
  - **offset**Type: Object助手（helper）的当前的绝对位置，表示为 `{ top, left }`。
  - **position**Type: Object助手（helper）的当前位置，表示为 `{ top, left }`.
  - **originalPosition**Type: Object元素的原始位置，表示为 `{ top, left }`.
  - **sender**Type: jQuery如果从一个 sortable 移动到另一个 sortable，项目来自的那个
  - **placeholder**Type: jQueryjQuery 对象，表示被作为占位符使用的元素。

Code examples:

使用指定的 sort 回调的 sortable：

```
$( ".selector" ).sortable({
  sort: function( event, ui ) {}
});
```

绑定一个事件监听器到 sort 事件：

```
$( ".selector" ).on( "sort", function( event, ui ) {} );
```

start(event, ui)Type: `sortstart`

当排序开始时触发该事件。

- **event**Type: Event
- **ui**Type: Object
  - **helper**Type: jQueryjQuery 对象，表示被排序的助手（helper）。
  - **item**Type: jQueryjQuery 对象，表示当前被拖拽的元素。
  - **offset**Type: Object助手（helper）的当前的绝对位置，表示为 `{ top, left }`。

- **position**Type: Object助手（helper）的当前位置，表示为 `{ top, left }`.
- **originalPosition**Type: Object元素的原始位置，表示为 `{ top, left }`.
- **sender**Type: jQuery如果从一个 sortable 移动到另一个 sortable，项目来自的那个
- **placeholder**Type: jQueryjQuery 对象，表示被作为占位符使用的元素。

Code examples:

使用指定的 start 回调的 sortable：

```
$( ".selector" ).sortable({
  start: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortstart 事件：

```
$( ".selector" ).on( "sortstart", function( event, ui ) {} );
```

stop(event, ui)Type: `sortstop`

当排序停止时触发该事件。

- **event**Type: Event
- **ui**Type: Object
    - **helper**Type: jQueryjQuery 对象，表示被排序的助手（helper）。
    - **item**Type: jQueryjQuery 对象，表示当前被拖拽的元素。
    - **offset**Type: Object助手（helper）的当前的绝对位置，表示为 `{ top, left }`。
    - **position**Type: Object助手（helper）的当前位置，表示为 `{ top, left }`.
    - **originalPosition**Type: Object元素的原始位置，表示为 `{ top, left }`.
    - **sender**Type: jQuery如果从一个 sortable 移动到另一个 sortable，项目来自的那个
    - **placeholder**Type: jQueryjQuery 对象，表示被作为占位符使用的元素。

Code examples:

使用指定的 stop 回调的 sortable：

```
$( ".selector" ).sortable({
  stop: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortstop 事件：

```
$( ".selector" ).on( "sortstop", function( event, ui ) {} );
```

update(event, ui)Type: `sortupdate`

当用户停止排序且 DOM 位置改变时触发该事件。

- **event**Type: Event
- **ui**Type: Object
  - **helper**Type: jQueryjQuery 对象，表示被排序的助手（helper）。
  - **item**Type: jQueryjQuery 对象，表示当前被拖拽的元素。
  - **offset**Type: Object助手（helper）的当前的绝对位置，表示为 `{ top, left }` 。
  - **position**Type: Object助手（helper）的当前位置，表示为 `{ top, left }` .
  - **originalPosition**Type: Object元素的原始位置，表示为 `{ top, left }` .
  - **sender**Type: jQuery如果从一个 sortable 移动到另一个 sortable，项目来自的那个
  - **placeholder**Type: jQueryjQuery 对象，表示被作为占位符使用的元素。

Code examples:

使用指定的 update 回调的 sortable：

```
$( ".selector" ).sortable({
  update: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortupdate 事件：

```
$( ".selector" ).on( "sortupdate", function( event, ui ) {} );
```

Example:

A simple jQuery UI Sortable.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>sortable demo</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
</head>
<body>

<ul id="sortable">
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
  <li>Item 5</li>
</ul>

<script>$("#sortable").sortable();</script>

</body>
</html>
```

Method Overrides

jQuery UI 重载了几个内置的 jQuery 方法，以提供额外的功能。当使用这些重载时，确保加载 jQuery UI 是很重要的。如果 jQuery UI 未加载，方法依然存在，但预期的功能将不可用，这会导致难以追踪的错误。

Also in: Effects | Effects Core

.addClass()

当动画样式改变时，为匹配的元素集合内的每个元素添加指定的 Class。

Also in: Methods | UI Core

.focus()

异步聚焦到一个元素。

Also in: Effects | Effects Core | Methods

.hide()

使用自定义效果来隐藏匹配的元素。

Also in: Methods | Utilities

.position()

相对另一个元素定位一个元素。

Also in: Effects | Effects Core

.removeClass()

当动画样式改变时，为匹配的元素集合内的每个元素移除指定的 Class。

Also in: Effects | Effects Core | Methods

.show()

使用自定义效果来显示匹配的元素。

Also in: Effects | Effects Core | Methods

.toggle()

使用自定义效果来显示或隐藏匹配的元素。

Also in: Effects | Effects Core

.toggleClass()

当动画样式改变时，根据 Class 是否存在以及 switch 参数的值，为匹配的元素集合内的每个元素添加或移除一个或多个 Class。

.focus()

Categories: Method Overrides | Methods | UI Core

.focus(delay [, callback])Returns: jQuery

Description: 异步聚焦到一个元素。

- **.focus(delay [, callback])**
  - **delay**Type: Integer聚焦前等待的毫秒数。
  - **callback**Type: Function()元素被聚焦后要调用的函数。

该插件扩展自 jQuery 内置的 `.focus()` 方法。如果 jQuery UI 未加载，调用 `.focus()` 方法不会直接失败，因为该方法在 jQuery 中存在。但是不会发生预期的行为。

.position()

Categories: Method Overrides | Methods | Utilities

.position(options)Returns: jQueryversion added: 1.8

Description: 相对另一个元素定位一个元素。

- .position(options)
    - **options**Type: Object
        - **my**（默认值：`"center"`） 类型：String 描述：定义被定位元素上对准目标元素的位置："horizontal vertical" 对齐方式。一个单一的值，比如 "right" 将规范为 "right center"，"top" 将规范为 "center top"（下面的 CSS 公约）。可接受的水平值："left"、"center"、"right"。可接受的垂直值："top"、"center"、"bottom"。例如，"left top" 或 "center center"。每个纬度也可以包含偏移，以像素计或以百分比计，例如 "right+10 top-25%"。百分比偏移是相对于被定位的元素。
        - **at**（默认值：`"center"`） 类型：String 描述：定义目标元素上对准被定位元素的位置： "horizontal vertical" 对齐方式。如需了解更多细节请查看 my 选项上的可能值。百分比偏移是相对于目标元素。
        - **of**（默认值：`null`） 类型：Selector 或 Element 或 jQuery 或 Event 描述：要定位的元素。如果您提供一个选择器（Selector）或 jQuery 对象，将使用第一个匹配元素。如果您提供一个事件（Event）对象，将使用 pageX 和 pageY 属性，例如 "#top-menu"。
        - **collision**（默认值：`"flip"`） 类型：String 描述：当被定位元素在某些方向上溢出窗口，则移动它到另一个位置。与 my 和 at 选项相似，该选项会接受一个单一的值或一对 horizontal/vertical 值。例如："flip"、"fit"、"fit flip"、"fit none"。
            - "flip"：翻转元素到目标的相对一边，再次运行 collision 检测一遍查看元素是否适合。无论哪一边允许更多的元素可见，则使用那一边。
            - "fit"：把元素从窗口的边缘移开。
            - "flipfit"：首先应用 flip 逻辑，把元素放置在允许更多元素可见的那一边。然后应用 fit 逻辑，确保尽可能多的元素可见。
            - "none"：不应用任何 collision 检测。
        - **using**（默认值：`null`） 类型：Function() 描述：当指定了该选项，实际属性设置则委托给该回调。接受两个参数：第一个是位置 top 和 left 值的哈希，可被转发到 .css() 或 .animate()；第二个提供了关于两个元素的位置和尺寸的反馈，

同时也计算它们的相对位置。target 和 element 都有下列属性：element、left、top、width、height。另外，还有 horizontal、vertical 和 important，提供了十二个可能的方向，如 { horizontal: "center", vertical: "left", important: "horizontal" }。

- **within**（默认值：`window`） 类型：Selector 或 Element 或 jQuery 描述：元素定位为 within，会影响 collision 检测。如果您提供了一个选择器（Selector）或 jQuery 对象，则使用第一个匹配的元素。

jQuery UI `.position()` 方法允许您相对窗体（window）、文档、另一个元素或指针（cursor）/鼠标（mouse）来定位一个元素，无需考虑相对父元素的偏移（offset）。

注释：jQuery UI 不支持定位隐藏元素。

这是一个独立的 jQuery 插件，且对其他 jQuery UI 组件没有依赖关系。

该插件扩展自 jQuery 内置的 .position() 方法。如果 jQuery UI 未加载，调用 `.position()` 方法不会直接失败，因为该方法在 jQuery 中存在。但是不会发生预期的行为。

Example:

一个简单的 **jQuery UI** 定位（**Position**）实例。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>position demo</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-ui.cs
  <style>
  .positionDiv {
    position: absolute;
    width: 75px;
    height: 75px;
    background: green;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="targetElement">
  <div class="positionDiv" id="position1"></div>
  <div class="positionDiv" id="position2"></div>
  <div class="positionDiv" id="position3"></div>
  <div class="positionDiv" id="position4"></div>
</div>

<script>
$( "#position1" ).position({
  my: "center",
  at: "center",
  of: "#targetElement"
});

$( "#position2" ).position({
  my: "left top",
  at: "left top",
  of: "#targetElement"
});

$( "#position3" ).position({
  my: "right center",
  at: "right bottom",
  of: "#targetElement"
});

$( document ).mousemove(function( event ) {
  $( "#position4" ).position({
    my: "left+3 bottom-3",
    of: event,
    collision: "fit"
  });
});
</script>

</body>
</html>
```

Methods

虽然jQuery UI主要包含的是widgets, interactions, h和 effects, 也添加了几个简单方便的方法。

Also in: UI Core

.disableSelection()

禁用选择匹配的元素集合内的文本内容。

Also in: Effects | Effects Core

.effect()

对一个元素应用动画特效。

Also in: UI Core

.enableSelection()

启用选择匹配的元素集合内的文本内容。

Also in: Method Overrides | UI Core

.focus()

异步聚焦到一个元素。

Also in: Effects | Effects Core | Method Overrides

.hide()

使用自定义效果来隐藏匹配的元素。

Also in: Method Overrides | Utilities

.position()

相对另一个元素定位一个元素。

Also in: UI Core

.removeUniqueId()

为匹配的元素集合移除由 .uniqueId() 设置的 Id。

Also in: UI Core

.scrollParent()

获取最近的可滚动的祖先。

Also in: Effects | Effects Core | Method Overrides

.show()

使用自定义效果来显示匹配的元素。

Also in: Effects | Effects Core | Method Overrides

.toggle()

使用自定义效果来显示或隐藏匹配的元素。

Also in: UI Core

.uniqueId()

为匹配的元素集合生成并申请一个唯一的 Id。

Also in: UI Core

.zIndex()

为元素获取 z-index。

.disableSelection()

Categories: Methods | UI Core

.disableSelection()Returns: jQuery

Description: 禁用选择匹配的元素集合内的文本内容。

- **version added: 1.6, deprecated: 1.9**.disableSelection()
    - 该方法不接受任何参数。

禁用的文本选择是有害的，不建议使用。

.enableSelection()

Categories: Methods | UI Core

.enableSelection()Returns: jQuery

Description: 启用选择匹配的元素集合内的文本内容。

- **version added: 1.6, deprecated: 1.9.enableSelection()**
  - 该方法不接受任何参数。

`.enableSelection()` 方法可用于启用通过 `.disableSelection()` 禁用的文本选择。

.removeUniqueId()

Categories: Methods | UI Core

.removeUniqueId()Returns: jQuery

Description: 为匹配的元素集合移除由 `.uniqueId()` 设置的 Id。

- **version added: 1.9.removeUniqueId()**
  - 该方法不接受任何参数。

`.removeUniqueId()` 移除由 `.uniqueId()` 设置的 id。在未使用 `.uniqueId()` 设置 id 的元素上调用 `.removeUniqueId()` 则无影响，即使该元素有一个 id。

.scrollParent()

Categories: Methods | UI Core

.scrollParent()Returns: jQuery

Description: 获取最近的可滚动的祖先。

- **.scrollParent()**
  - 该方法不接受任何参数。

该方法查找最近的可滚动的祖先。换句话说，`.scrollParent()` 查找当前所选元素在其内滚动的元素。

注意*:* 该方法只在包含一个元素的 *jQuery* 对象上工作。

.uniqueId()

Categories: Methods | UI Core

.uniqueId()Returns: jQuery

Description: 为匹配的元素集合生成并申请一个唯一的 Id。

- **version added: 1.9.uniqueId()**
  - 该方法不接受任何参数。

许多小部件需要元素生成唯一的 id。 `.uniqueId()` 会检查元素是否有 id，如果元素没有 id，它将生成一个 id，并设置为该元素的 id。在未检查元素是否具有 id 就调用 `.uniqueId()` 是安全的。当小部件使用后需要清除，如果 id 是通过 `.uniqueId()` 添加的， `.removeUniqueId()` 方法将从元素上移除 id，如果 id 不是通过 .uniqueId() 添加的，则无影响。 `.removeUniqueId()` 之所以能区分 id，是因为 `.uniqueId()` 生成的 id 带有前缀 "ui-id-"。

.zIndex()

Categories: Methods | UI Core

.zIndex()Returns: jQuery

Description: 为元素获取 z-index。

- **.zIndex()**
  - 该方法不接受任何参数。

`.zIndex()` 方法在查找一个元素的 z-index 时非常有用，忽略 z-index 是否是直接设置在元素上还是设置在祖先元素上。为了确定 z-index，该方法会在指定的元素上开始，且会沿着 DOM 查找，直到找到一个带有 z-index 的已定位的元素。如果未找到这样的元素，该方法将返回一个 `0` 值。

该方法假设带有嵌套 z-index 的元素不带有一个 `0` 值的 z-index。例如，给出下面的 DOM，内部元素将被当成不带有 z-index，因为在 Internet Explorer 中无法区分一个 `0` 显式值和无值。

```
<div style="z-index: -10;">
  <div style="z-index: 0;"></div>
</div>
```

.zIndex(zIndex)Returns: Integer

Description: 为元素设置 z-index。

- **.zIndex(zIndex)**
  - **zIndex**Type: Integer要设置的 z-index。

这相当于 `.css( "zIndex", zIndex )`。

Selectors

:data() Selector

选择数据已存储在指定的键下的元素。

Also in: UI Core

:focusable Selector

选择可被聚焦的元素。

Also in: UI Core

:tabbable Selector

选择用户可通过 tab 键聚焦的元素。

:data() Selector

Categories: Selectors | UI Core

Description: 选择数据已存储在指定的键下的元素。

- **jQuery(":data(key)")**

  key: 数据的键。

表达式 `$( "div:data(foo)")` 匹配一个通过 `.data( "foo", value )` 存储数据的 `<div>` 。

Example:

选择带有数据的元素，改变它们的背景颜色。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>data demo</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-ui.cs
  <style>
  div {
    width: 100px;
    height: 100px;
    border: 1px solid #000;
    float: left;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="one"></div>
<div id="two"></div>
<div id="three"></div>

<script>
$( "#one" ).data( "color", "blue" );
$( "#three" ).data( "color", "green" );

$( ":data(color)" ).each(function() {
  var element = $( this );
  element.css( "backgroundColor", element.data( "color" ) );
});
</script>

</body>
</html>
```

:focusable Selector

Categories: Selectors | UI Core

Description: 选择可被聚焦的元素。

- **jQuery(":focusable")**

一些元素本身是可聚焦的（focusable），而另一些元素需要显式设置 tab 索引。以上两种情况，为了可聚焦（focusable），元素都必须是可见的。

下面类型的元素如果未被禁用，则是可聚焦的（focusable）：`input`、`select`、`textarea`、`button` 和 `object`。锚如果带有 `href` 或 `tabindex` 属性，则是可聚焦的（focusable）。`area` 元素如果在一个已命名的 map 内，且带有 `href` 属性，且有一个可见的图像使用了该 map，则是可聚焦的（focusable）。所有其他完全基于 `tabindex` 属性和可见度的元素是可聚焦的（focusable）。

注释：带有负的 tab 索引的元素是 `:focusable`，不是 `[:tabbable](/tabbable-selector/)`。

Example:

选择可聚焦的元素，且用一个红色边框突出显示。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>focusable demo</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-ui.cs
  <style>
  input, a, p {
    border: 1px solid #000;
  }
  div {
    padding: 5px;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div><input value="text input"></div>
<div><a>anchor without href</a></div>
<div><a href="#">anchor with href</a></div>
<div><p>paragraph without tabindex</p></div>
<div><p tabindex="1">paragraph with tabindex</p></div>

<script>
$( ":focusable" ).css( "border-color", "red" );
</script>

</body>
</html>
```

:tabbable Selector

Categories: Selectors | UI Core

Description: 选择用户可通过 tab 键聚焦的元素。

- **jQuery(":tabbable")**

一些元素本身是可通过 tab 键聚焦的（tabbable），而另一些元素需要显式设置一个正的 tab 索引。以上两种情况，为了可通过 tab 键聚焦（tabbable），元素都必须是可见的。

下面类型的元素如果不带有负的 tab 索引且未被禁用，则是可通过 tab 键聚焦的（tabbable）：`input`、`select`、`textarea`、`button` 和 `object`。锚如果带有 `href` 或正的 `tabindex` 属性，则是可通过 tab 键聚焦的（tabbable）。`area` 元素如果在一个已命名的 map 内，且带有 `href` 属性，且有一个可见的图像使用了该 map，则是可通过 tab 键聚焦的（tabbable）。所有其他完全基于 `tabindex` 属性和可见度的元素是可通过 tab 键聚焦的（tabbable）。

注释：带有负的 tab 索引的元素是 `:focusable`，不是 `:tabbable`。

Example:

选择可通过 **tab** 键聚焦的元素，且用一个红色边框突出显示。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>tabbable demo</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-ui.cs
  <style>
  input {
    border: 1px solid #000;
  }
  div {
    padding: 5px;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div><input value="no tabindex"></div>
<div><input tabindex="5" value="positive tabindex"></div>
<div><input tabindex="-1" value="negative tabindex"></div>

<script>
$( ":tabbable" ).css( "border-color", "red" );
</script>

</body>
</html>
```

Theming

jQuery UI 包括一个强大的 CSS 框架，用于创建自定义的 jQuery 小部件。该框架包括涵盖广泛的公共用户界面需求的 Class，并且可以使用 jQuery UI ThemeRoller 进行操纵。通过使用 jQuery UI CSS 框架创建您自己的 UI 组件，您应采用共享标记公约，便于插件社区的代码集成。您可以查看 jQuery UI 主题 了解更多详情。

CSS Framework

jQuery UI 使用的允许组件主题化的 Class 名称列表。

Icons

jQuery UI 提供的图标列表。

Stacking Elements

一种处理 z-index 和堆叠元素的模式。

CSS 框架（CSS Framework）

下面是 jQuery UI 使用的 Class 名称列表。这些 Class 用来创建跨应用程序的视觉一致性，且允许组件通过 jQuery UI ThemeRoller 进行主题化。下面的 CSS 类根据样式是否是固定的结构化的，或者是否是可主题化的（颜色、字体、背景等），分别定义在 ui.core.css 和 ui.theme.css 两个文件中。

布局助手

- `.ui-helper-hidden` ：对元素应用 `display: none` 。
- `.ui-helper-hidden-accessible` ：对元素应用访问隐藏（通过页面绝对定位）。
- `.ui-helper-reset` ：UI 元素的基本样式重置。重置的元素比如： `padding` 、 `margin` 、 `text-decoration` 、list-style，等等。
- `.ui-helper-clearfix` ：对父元素应用浮动包装属性。
- `.ui-helper-zfix` ：对 `<iframe>` 元素应用 iframe "fix" CSS。
- `.ui-front` ：应用 z-index 来管理屏幕上多个小部件的堆叠，如需了解更多详情，请查看 堆叠元素（Stacking Elements） 页面。

小部件容器

- `.ui-widget` ：对所有小部件的外部容器应用的 Class。对小部件应用字体和字体尺寸，同时也对自表单元素应用相同的字体和 1em 的字体尺寸，以应对 Windows 浏览器中的继承问题。
- `.ui-widget-header` ：对标题容器应用的 Class。对元素及其子元素的文本、链接、图标应用标题容器样式。
- `.ui-widget-content` ：对内容容器应用的 Class。对元素及其子元素的文本、链接、图标应用内容容器样式。（可应用到标题的父元素或者同级元素）

交互状态

- `.ui-state-default` ：对可点击按钮元素应用的 Class。对元素及其子元素的文本、链接、图标应用 "clickable default" 容器样式。
- `.ui-state-hover` ：当鼠标悬浮在可点击按钮元素上时应用的 Class。对元素及其子元素的文本、链接、图标应用 "clickable hover" 容器样式。
- `.ui-state-focus` ：当键盘聚焦在可点击按钮元素上时应用的 Class。对元素及其子元素的文本、链接、图标应用 "clickable hover" 容器样式。
- `.ui-state-active` ：当鼠标点击可点击按钮元素上时应用的 Class。对元素及其子元素的文本、链接、图标应用 "clickable active" 容器样式。

交互提示 Cues

- `.ui-state-highlight` ：对高亮或者选中元素应用的 Class。对元素及其子元素的文本、链接、图标应用 "highlight" 容器样式。
- `.ui-state-error` ：对错误消息容器元素应用的 Class。对元素及其子元素的文本、链接、图标应用 "error" 容器样式。
- `.ui-state-error-text` ：对只有无背景的错误文本颜色应用的 Class。可用于表单标签，也可以对子图标应用错误图标颜色。
- `.ui-state-disabled` ：对禁用的 UI 元素应用一个暗淡的不透明度。意味着对一个已经定义样式的元素添加额外的样式。
- `.ui-priority-primary` ：对第一优先权的按钮应用的 Class。应用粗体文本。
- `.ui-priority-secondary` ：对第二优先权的按钮应用的 Class。应用正常粗细的文本，对元素应用轻微的透明度。

图标

状态和图像

- `.ui-icon` ：对图标元素应用的基本 Class。设置尺寸为 16px 方块，隐藏内部文本，对 "content" 状态的精灵图像设置背景图像。注意： *`.ui-icon` class* 将根据它的父容器得到一个不同的精灵背景图像。例如， *`ui-state-default`* 容器内的 *`ui-icon`* 元素将根据 *`ui-state-default`* 的图标颜色进行着色。

图标类型

在声明 `.ui-icon` class 之后，接着您可以声明一个秒速图标类型的 class。通常情况下，图标 class 遵循语法 `.ui-icon-{icon type}-{icon sub description}-{direction}` 。

例如，一个指向右侧的三角形图标，如下所示： `.ui-icon-triangle-1-e`

jQuery UI 的 ThemeRoller 在它的预览一栏中提供了全套的 CSS 框架图标。将鼠标悬浮在图标上可查看 class 名称。

其他视觉效果

圆角半径助手

- `.ui-corner-tl` ：对元素的左上角应用圆角半径。
- `.ui-corner-tr` ：对元素的右上角应用圆角半径。
- `.ui-corner-bl` ：对元素的左下角应用圆角半径。
- `.ui-corner-br` ：对元素的右下角应用圆角半径。
- `.ui-corner-top` ：对元素上边的左右角应用圆角半径。
- `.ui-corner-bottom` ：对元素下边的左右角应用圆角半径。

- `.ui-corner-right` ：对元素右边的上下角应用圆角半径。
- `.ui-corner-left` ：对元素左边的上下角应用圆角半径。
- `.ui-corner-all` ：对元素的所有四个角应用圆角半径。

覆盖 & 阴影

- `.ui-widget-overlay` ：对覆盖屏幕应用 100% 宽度和高度，同时设置背景颜色/纹理和屏幕不透明度。
- `.ui-widget-shadow` ：对覆盖应用的 Class，设置了不透明度、上偏移/左偏移，以及阴影的 "厚度"。厚度是通过对阴影所有边设置内边距（padding）进行应用的。偏移是通过设置上外边距（margin）和左外边距（margin）进行应用的（可以是正数，也可以是负数）。

Icons

jQuery UI 提供了大量可以通过对元素应用 Class 名称来使用的图标（Icons）。Class 名称大体上遵循语法 `.ui-icon-{icon type}-{icon sub description}-{direction}` 。例如，下面将显示一个向北的厚箭头的图标：

```
<span class="ui-icon ui-icon-arrowthick-1-n"></span>
```

图标也集成到一些 jQuery UI 的小部件，比如 accordion、button、menu。

下面是 jQuery UI 提供的图标的完整列表：

Stacking Elements

堆叠的或者移动到其他元素前面的小部件（Widgets）当放置到现实世界的页面中时经常面临挑战。通常通过简单地改变堆叠元素的 `z-index` 或者父元素来避免页面上的冲突。但是，jQuery UI 需要一个不需要手动改变 `z-index` 值的通用的解决方案。这是通过 `ui-front` class 来完成的，通常还伴随着堆叠组件上的 `appendTo` 选项。

link The `ui-front` class

`ui-front` class 是非常基础的。它只是在元素上设置了一个静态的 `z-index` 值。但是，class 的存在是用来表明堆叠元素要追加到哪里。这允许我们利用嵌套层内容，生成一个在大多数情况下都能使用的默认的 DOM 位置。

注释：当使用 *`ui-front`* 时，您必须设置 *`position`* 为 *`relative`*、 *`absolute`* 或 *`fixed`*，以便应用 *`z-index`*。

link 堆叠技术（stacking technique）

追加堆叠元素到页面的任何小部件都必须使用 `ui-front` class，且在大多数情况下，应该有一个 `appendTo` 选项。堆叠元素应遵循下面的规则：

- 如果一个值设置为 `appendTo` 选项，则追加堆叠元素到指定的元素。
- 如果 `appendTo` 选项被设置为 `null`（默认），则小部件应从相关的元素开始遍历 DOM。例如，当自动完成（autocomplete）菜单被追加到 DOM，遍历则从相关的 input 元素开始。
    - 如果找到一个带有 `ui-front` class 的元素，则追加到该元素。
    - 如果没有找到一个带有 `ui-front` class 的元素，则追加到主体（body）。
- 堆叠元素的 `position` 必须设置为 `relative`、 `absolute` 或 `fixed`，以便应用来自 `ui-front` class 的 `z-index`。使用 .position() 将自动设置 `position`。

UI Core

由 jquery.ui.core.js 提供的功能。

Also in: Selectors

:data() Selector

选择数据已存储在指定的键下的元素。

Also in: Methods

.disableSelection()

禁用选择匹配的元素集合内的文本内容。

Also in: Methods

.enableSelection()

启用选择匹配的元素集合内的文本内容。

Also in: Method Overrides | Methods

.focus()

异步聚焦到一个元素。

Also in: Selectors

:focusable Selector

选择可被聚焦的元素。

jQuery.ui.keyCode

一个相对于数字值的关键代码描述的映射。

Also in: Methods

.removeUniqueId()

为匹配的元素集合移除由 .uniqueId() 设置的 Id。

Also in: Methods

.scrollParent()

获取最近的可滚动的祖先。

Also in: Selectors

:tabbable Selector

选择用户可通过 tab 键聚焦的元素。

Also in: Methods

.uniqueId()

为匹配的元素集合生成并申请一个唯一的 Id。

Also in: Methods

.zIndex()

为元素获取 z-index。

Utilities

Easings

Easing 函数指定动画在不同点上的行进速度。

Also in: Widgets

Widget Factory

使用与所有 jQuery UI 小部件相同的抽象化来创建有状态的 jQuery 插件。

Also in: Widgets

Widget Plugin Bridge

jQuery.widget.bridge() 方法是 jQuery 部件库（Widget Factory）的一部分。它扮演着由 $.widget() 创建的对象和 jQuery API 之间的中介。

Also in: Interactions

Mouse Interaction

基本交互层。

Also in: Method Overrides | Methods

.position()

相对另一个元素定位一个元素。

Widget Factory

Categories: Utilities | Widgets

jQuery.widget(name [, base], prototype)

Description: 使用与所有 jQuery UI 小部件相同的抽象化来创建有状态的 jQuery 插件。

- **jQuery.widget(name [, base], prototype)**
    - **name**Type: String要创建的小部件名称，包括命名空间。
    - **base**Type: Function()要继承的基础小部件。必须是一个可以使用 `new` 关键词实例化的构造函数。默认为 `jQuery.Widget` 。
    - **prototype**Type: PlainObject要作为小部件原型使用的对象。

您可以使用 `$.Widget` 对象作为要继承的基础，或者可以明确地从现有的 jQuery UI 或第三方控件，从头开始创建新的小部件。定义一个带有相同名称的小部件来继承基础部件，甚至允许您适当地扩展小部件。

jQuery UI 中包含许多保持状态的小部件，因此比典型的 jQuery 插件稍有不同的使用模式。所有的jQuery UI 小部件使用相同的模式，这是由部件库（Widget Factory）定义的。所以，只要您学会使用其中一个，您就知道如何使用其他的小部件（Widget）。

寻找有关小部件工厂的教程？查看jQuery学习中心的文章。

注意：本章节使用 进度条部件（Progressbar Widget） 演示实例，但是语法适用于每个小部件。

初始化

为了跟踪小部件的状态，我们必须引入小部件的全生命周期。小部件初始化时生命周期开始。要初始化一个小部件，我们只需要简单地在一个或多个元素上调用插件。

```
$( "#elem" ).progressbar();
```

这将初始化 jQuery 对象中的每个元素。上面实例中元素 id 为 `"elem"` 。

选项

由于 `progressbar()` 调用时不带参数，小部件是使用默认选项进行初始化的。我们可以在初始化时传递一组选项来覆盖默认选项：

```
$( "#elem" ).progressbar({ value: 20 });
```

我们可以根据需要传递选项的个数，任何我们未传递的选项都使用它们的默认值。

您可以传递多个选项参数，这些参数将会被合并为一个对象（类似于 `$.extend( true, target, object1, objectN )`）。这在为所有实例覆盖一些设置，实例间共享选项时很有用：

```
var options = { modal: true, show: "slow" };
$( "#dialog1" ).dialog( options );
$( "#dialog2" ).dialog( options, { autoOpen: false });
```

所有在初始化时传递的选项都是深拷贝的，确保后续在不影响小部件的情况下修改对象。数组是唯一的例外，它们是按原样引用的。这个例外是为了适当地支持数据绑定，其中数据源必须作为引用。

默认值保存在小部件的属性中，因此我们可以覆盖 jQuery UI 设置的值。例如，在下面的设置后，所有将来的进度条实例将默认为值 80：

```
$.ui.progressbar.prototype.options.value = 80;
```

选项是小部件状态的组成部分，所以我们也可以在初始化后设置选项。我们会在后续看到 option 方法。

方法

现在小部件已经初始化，我们可以查询它的状态，或者在小部件上执行动作。所有初始化后的动作都是以方法调用方式执行。为了在小部件上调用一个方法，我们向 jQuery 插件传递方法的名称。例如，在进度条部件（Progressbar Widget）上调用 `value()` 方法，我们可以使用：

```
$( "#elem" ).progressbar( "value" );
```

如果方法接受参数，我们可以在方法名称后传递参数。例如，要传递参数 `40` 到 `value()` 方法，我们可以使用：

```
$( "#elem" ).progressbar( "value", 40 );
```

就像 jQuery 中的其他方法，大多数的小部件方法返回 jQuery 对象：

```
$( "#elem" )
  .progressbar( "value", 90 )
  .addClass( "almost-done" );
```

每个小部件都有自己的方法设置，这些设置是基于小部件提供的功能。但是，有一些方法是存在于所有的小部件上，这会在下面进行详细讲解。

事件

所有的小部件都有与它们各种行为相关的事件，以便在状态改变的时候通知您。对于大多数的小部件，当事件被触发时，名称以小部件名称的小写字母形式作为前缀。例如，我们可以绑定进度条的 `change` 事件，该事件在值改变时触发。

```
$( "#elem" ).bind( "progressbarchange", function() {
  alert( "The value has changed!" );
});
```

每个事件都有一个对应的回调，这会作为选项。如果需要，我们可以抓住进度条的 `change` 回调，而不用绑定 `progressbarchange` 事件。

```
$( "#elem" ).progressbar({
  change: function() {
    alert( "The value has changed!" );
  }
});
```

所有的小部件都有一个 `change` 事件，该事件在实例化时触发。

实例化

小部件的实例是使用带有小部件全称作为键的 `jQuery.data()` 存储的。因此，您可以使用下面代码从元素检索进度条部件（Progressbar Widget）的实例对象。

```
$( "#elem" ).data( "ui-progressbar" );
```

元素是否绑定了给定小部件，可以使用 `:data` 选择器来检测。

```
$( "#elem" ).is( ":data( 'ui-progressbar' )" ); // true
$( "#elem" ).is( ":data( 'ui-draggable' )" ); // false
```

您也可以使用 `:data` 来获得作为给定小部件实例的所有元素的列表。

```
$( ":data( 'ui-progressbar' )" );
```

属性

所有的小部件都有下面的属性：

- **defaultElement**：当构造小部件实例未提供元素时要使用的元素。例如，由于进度条的 `defaultElement` 是 `"<div>"`，`$.ui.progressbar({ value: 50 })` 在一个新建的 `<div>` 上实例化进度条小部件实例。
- **document**：其内包含小部件元素的 `document`。如果需要在框架内与小部件进行交互时很有用。
- **element**：一个 jQuery 对象，包含用于实例化小部件的 元素。如果您选择多个元素并调用 `.myWidget()`，将为每个元素创建一个单独的小部件实例。因此，该属性总是包含一个元素。
- **namespace**：小部件原型存储的全局 jQuery 对象的位置。例如，`"ui"` 的 `namespace` 表示小部件原型存储在 `$.ui`。
- **options**：一个包含小部件当前使用选项的对象。在实例化时，用户提供的任何选项将会自动与 `$.myNamespace.myWidget.prototype.options` 中定义的默认值合并。用户指定的选项会覆盖默认值。
- **uuid**：一个表示控件标识符的唯一整数。
- **version**：小部件的字符串版本。对于 jQuery UI 小部件，该属性会被设置为小部件使用的 jQuery UI 的版本。插件开发者必须在他们的原型中明确设置该属性。
- **widgetEventPrefix**：添加到小部件事件名称的前缀。例如，可拖拽小部件（Draggable Widget）的 `widgetEventPrefix` 是 `"drag"`，因此当创建一个 draggable 时，事件的名称是 `"dragcreate"`。默认情况下，小部件的 `widgetEventPrefix` 是它的名称。注意：该属性已被废弃，将在以后的版本中非常。事件名称将被改为 *widgetName:eventName*（例如 *`"draggable:create"`*）。
- **widgetFullName**：包含命名空间的小部件全称。对于 `$.widget( "myNamespace.myWidget", {} )`，`widgetFullName` 将是 `"myNamespace-myWidget"`。
- **widgetName**：小部件的名称。对于 `$.widget( "myNamespace.myWidget", {} )`，`widgetName` 将是 `"myWidget"`。
- **window**：其内包含小部件元素的 `window`。如果需要在框架内与小部件进行交互时很有用。

Description: 部件库（Widget Factory）使用的基础小部件。

QuickNav

Options

- disabled
- hide
- show

Methods

- _create
- _delay
- _destroy
- _focusable
- _getCreateEventData
- _getCreateOptions
- _hide
- _hoverable
- _init
- _off
- _on
- _setOption
- _setOptions
- _show
- _super
- _superApply
- _trigger
- destroy
- disable
- enable
- option
- widget

Events

- create

Options

disabledType: Boolean

Default: `false` 如果设置为 `true`，则禁用该小部件。**Code examples:**

初始化带有指定 `disabled` 选项的小部件：

```
$( ".selector" ).widget({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).widget( "option", "disabled" );

// setter
$( ".selector" ).widget( "option", "disabled", true );
```

hideType: Boolean or Number or String or Object

Default: `null` 是否使用动画隐藏元素，以及如何动画隐藏元素。支持多个类型：

- **Boolean**：当设置为 `false` 时，则不使用动画，元素会立即隐藏。当设置为 `true` 时，元素会使用默认的持续时间和默认的 easing 淡出。
- **Number**：元素将使用指定的持续时间和默认的 easing 淡出。
- **String**：元素将使用指定的特效隐藏。该值可以是一个内建的 jQuery 动画方法名称，比如 `"slideUp"`，也可以是一个 jQuery UI 特效 的名称，比如 `"fold"`。以上两种情况的特效将使用默认的持续时间和默认的 easing。
- **Object**：如果值是一个对象，则 `effect`、`delay`、`duration` 和 `easing` 属性会被提供。如果 `effect` 属性包含 jQuery 方法的名称，则使用该方法，否则，它被认为是一个 jQuery UI 特效的名称。当使用一个支持额外设置的 jQuery UI 特效时，您可以在对象中包含这些设置，且它们会被传递给特效。如果 `duration` 或 `easing` 被省略，则使用默认值。如果 `effect` 被省略，则使用 `"fadeOut"`。如果 `delay` 被省略，则不使用延迟。

Code examples:

初始化带有指定 `hide` 选项的小部件：

```
$( ".selector" ).widget({ hide: { effect: "explode", duration: 1000 } });
```

在初始化后，获取或设置 `hide` 选项：

```
// getter
var hide = $( ".selector" ).widget( "option", "hide" );

// setter
$( ".selector" ).widget( "option", "hide", { effect: "explode", duration: 1000 } );
```

showType: Boolean or Number or String or Object

Default: `null` 是否使用动画显示元素，以及如何动画显示元素。支持多个类型：

- **Boolean**：当设置为 `false` 时，则不使用动画，元素会立即显示。当设置为 `true` 时，元素会使用默认的持续时间和默认的 easing 淡入。
- **Number**：元素将使用指定的持续时间和默认的 easing 淡入。
- **String**：元素将使用指定的特效显示。该值可以是一个内建的 jQuery 动画方法名称，比如 `"slideDown"`，也可以是一个 jQuery UI 特效 的名称，比如 `"fold"`。以上两种情况的特效将使用默认的持续时间和默认的 easing。
- **Object**：如果值是一个对象，则 `effect`、`delay`、`duration` 和 `easing` 属性会被提供。如果 `effect` 属性包含 jQuery 方法的名称，则使用该方法，否则，它被认为是一个 jQuery UI 特效的名称。当使用一个支持额外设置的 jQuery UI 特效时，您可以在对象中包含这些设置，且它们会被传递给特效。如果 `duration` 或 `easing` 被省略，则使用默认值。如果 `effect` 被省略，则使用 `"fadeIn"`。如果 `delay` 被省略，则不使用延迟。

Code examples:

初始化带有指定 `show` 选项的小部件：

```
$( ".selector" ).widget({ show: { effect: "blind", duration: 800 } });
```

在初始化后，获取或设置 `show` 选项：

```
// getter
var show = $( ".selector" ).widget( "option", "show" );

// setter
$( ".selector" ).widget( "option", "show", { effect: "blind", duration: 800 } );
```

Methods

_create()Returns: jQuery (plugin only)

`_create()` 方法是小部件的构造函数。没有参数，但是 `this.element` 和 `this.options` 已经设置。

- 该方法不接受任何参数。

Code examples:

基于一个选项设置小部件元素的背景颜色。

```
_create: function() {
  this.element.css( "background-color", this.options.color );
}
```

_delay(fn [, delay])Returns: Number

在指定延迟后调用提供的函数。保持 `this` 上下文正确。本质上是 `setTimeout()`。

使用 `clearTimeout()` 返回超时 ID。

- **fn**Type: Function() or String要调用的函数。也可以是小部件上方法的名称。
- **delay**Type: Number调用函数前等待的毫秒数，默认为 `0`。

Code examples:

100 毫秒后在小部件上调用 `_foo()` 方法。

```
this._delay( this._foo, 100 );
```

_destroy()Returns: jQuery (plugin only)

公共的 `destroy()` 方法清除所有的公共数据、事件等等。代表了定制、指定小部件、清理的 `_destroy()`。

- 该方法不接受任何参数。

Code examples:

当小部件被销毁时，从小部件的元素移除一个 class。

```
_destroy: function() {
  this.element.removeClass( "my-widget" );
}
```

_focusable(element)Returns: jQuery (plugin only)

建立聚焦在元素上时要应用 `ui-state-focus` 的 `element`。

事件处理程序在销毁时自动清理。

- **element**Type: jQuery要应用 focusable 行为的元素。

Code examples:

向小部件内的一组元素应用 focusable 样式：

```
this._focusable( this.element.find( ".my-items" ) );
```

_getCreateEventData()Returns: Object

所有的小部件触发 `create` 事件。默认情况下，事件中不提供任何的数据，但是该方法会返回一个对象，作为 `create` 事件的数据被传递。

- 该方法不接受任何参数。

Code examples:

向 `create` 事件处理程序传递小部件的选项，作为参数。

```
_getCreateEventData: function() {
  return this.options;
}
```

_getCreateOptions()Returns: Object

该方法允许小部件在初始化期间为定义选项定义一个自定义的方法。用户提供的选项会覆盖该方法返回的选项，即会覆盖默认的选项。

- 该方法不接受任何参数。

Code examples:

让小部件元素的 id 属性作为选项可用。

```
_getCreateOptions: function() {
  return { id: this.element.attr( "id" ) };
}
```

_hide(element, option [, callback])Returns: jQuery (plugin only)

使用内置的动画方法或使用自定义的特效隐藏一个元素。如需了解可能的 `option` 值，请查看 hide。

- **element**Type: jQuery要隐藏的元素。
- **option**Type: Object定义如何隐藏元素的设置。
- **callback**Type: Function()元素完全隐藏后要调用的回调。

Code examples:

为自定义动画传递 `hide` 选项。

```
this._hide( this.element, this.options.hide, function() {

  // Remove the element from the DOM when it's fully hidden.
  $( this ).remove();
});
```

_hoverable(element)Returns: jQuery (plugin only)

建立悬浮在元素上时要应用 `ui-state-hover` class 的 `element` 。

事件处理程序在销毁时自动清理。

- **element**Type: jQuery要应用 hoverable 行为的元素。

Code examples:

当悬浮在元素上时，向元素内所有的 `<div>` 应用 hoverable 样式。

```
this._hoverable( this.element.find( "div" ) );
```

_init()Returns: jQuery (plugin only)

小部件初始化的理念与创建不同。任何时候不带参数的调用插件或者只带一个选项哈希的调用插件，初始化小部件。当小部件被创建时会包含这个方法。

注意：如果存在不带参数成功调用小部件时要执行的逻辑动作，初始化只能在这时处理。

- 该方法不接受任何参数。

Code examples:

如果设置了 `autoOpen` 选项，则调用 `open()` 方法。

```
_init: function() {
  if ( this.options.autoOpen ) {
    this.open();
  }
}
```

_off(element, eventName)Returns: jQuery (plugin only)

从指定的元素取消绑定事件处理程序。

- **element**Type: jQuery要取消绑定事件处理程序的元素。不像 `_on()` 方法，`_off()` 方法中元素是必需的。
- **eventName**Type: String一个或多个空格分隔的事件类型。

Code examples:

从小部件的元素上取消绑定所有 click 事件。

```
this._off( this.element, "click" );
```

_on([suppressDisabledCheck] [, element], handlers)Returns: jQuery (plugin only)

授权通过事件名称内的选择器被支持，例如 "`click .foo`"。 `_on()` 方法提供了一些直接事件绑定的好处：

- 保持处理程序内适当的 `this` 上下文。
- 自动处理禁用的部件：如果小部件被禁用或事件发生在带有 `ui-state-disabled` class 的元素上，则不调用事件处理程序。可以被 `suppressDisabledCheck` 参数重写。
- 事件处理程序会自动添加命名空间，在销毁时会自自动清理。

- **suppressDisabledCheck** (default: `false`)Type: Boolean是否要绕过禁用的检查。

- **element**Type: jQuery要绑定事件处理程序的元素。如果未提供元素， `this.element` 用于未授权的事件， widget 元素 用于授权的事件。
- **handlers**Type: Object一个 map，其中字符串键代表事件类型，可选的选择器用于授权，值代表事件调用的处理函数。

Code examples:

放置小部件元素内所有被点击的链接的默认行为。

```
this._on( this.element, {
  "click a": function( event ) {
    event.preventDefault();
  }
});
```

_setOption(key, value)Returns: jQuery (plugin only)

为每个独立的选项调用 `_setOptions()` 方法。小部件状态随着改变而更新。

- **key**Type: String要设置的选项名称。
- **value**Type: Object为选项设置的值。

Code examples:

当小部件的 `height` 或 `width` 选项改变时，更新小部件的元素。

```
_setOption: function( key, value ) {
  if ( key === "width" ) {
    this.element.width( value );
  }
  if ( key === "height" ) {
    this.element.height( value );
  }
  this._super( key, value );
}
```

_setOptions(options)Returns: jQuery (plugin only)

当调用 `option()` 方法时调用，无论以什么形式调用 `option()` 。

如果您要根据多个选项的改变而改变处理器密集型，重载该方法是很有用的。

- **options**Type: Object要设置的选项-值对。

Code examples:

如果小部件的 `height` 或 `width` 选项改变，调用 `resize()` 方法。

```
_setOptions: function( options ) {
  var that = this,
    resize = false;

  $.each( options, function( key, value ) {
    that._setOption( key, value );
    if ( key === "height" || key === "width" ) {
      resize = true;
    }
  });

  if ( resize ) {
    this.resize();
  }
}
```

_show(element, option [, callback])Returns: jQuery (plugin only)

使用内置的动画方法或使用自定义的特效显示一个元素。如需了解可能的 `option` 值，请查看 show。

- **element**Type: jQuery要显示的元素。
- **option**Type: Object定义如何显示元素的设置。
- **callback**Type: Function()元素完全显示后要调用的回调。

Code examples:

为自定义动画传递 `show` 选项。

```
this._show( this.element, this.options.show, function() {

  // Focus the element when it's fully visible.
  this.focus();
}
```

_super([arg] [, ...])Returns: jQuery (plugin only)

从父部件中调用相同名称的方法，带有任意指定的参数。本质上是 `.call()` 。

- **arg**Type: Object要传递给父部件的方法的零到多个参数。

Code examples:

处理 `title` 选项更新，并调用付部件的 `_setOption()` 来更新选项的内部存储。

```
_setOption: function( key, value ) {
  if ( key === "title" ) {
    this.element.find( "h3" ).text( value );
  }
  this._super( key, value );
}
```

_superApply(arguments)Returns: jQuery (plugin only)

从父部件中调用相同名称的方法，带有参数的数组。本质上是 `.apply()` 。

- **arguments**Type: Array要传递给父部件的方法的参数数组。

Code examples:

处理 `title` 选项更新，并调用付部件的 `_setOption()` 来更新选项的内部存储。

```
_setOption: function( key, value ) {
  if ( key === "title" ) {
    this.element.find( "h3" ).text( value );
  }
  this._superApply( arguments );
}
```

_trigger(type [, event] [, data])Returns: Boolean

触发一个事件及其相关的回调。

带有该名称的选项与作为回调被调用的类型相等。

事件名称是小部件名称和类型的小写字母串。

注意: 当提供数据时，您必须提供所有三个参数。如果没有传递事件，则传递 *`null`* 。

如果默认行为是阻止的，则返回 `false` ，否则返回 `true` 。当处理程序返回 `false` 时或调用 `event.preventDefault()` 时，则阻止默认行为发生。

- **type**Type: StringThe `type` 应该匹配回调选项的名称。完整的事件类型会自动生成。
- **event**Type: Event导致该事件发生的原始事件，想听众提供上下文时很有用。
- **data**Type: Object一个与事件相关的数据哈希。

Code examples:

当按下一个键时，触发 `search` 事件。

```
this._on( this.element, {
  keydown: function( event ) {

    // Pass the original event so that the custom search event has
    // useful information, such as keyCode
    this._trigger( "search", event, {

      // Pass additional information unique to this event
      value: this.element.val()
    });
  }
});
```

destroy()Returns: jQuery (plugin only)

完全移除小部件功能。这会把元素返回到它的预初始化状态。

- 该方法不接受任何参数。

Code examples:

当点击小部件的任意锚点时销毁小部件。

```
this._on( this.element, {
  "click a": function( event ) {
    event.preventDefault();
    this.destroy();
  }
});
```

disable()Returns: jQuery (plugin only)

禁用小部件。

- 该方法不接受任何参数。

Code examples:

当点击小部件的任意锚点时禁用小部件。

```
this._on( this.element, {
  "click a": function( event ) {
    event.preventDefault();
    this.disable();
  }
});
```

enable()Returns: jQuery (plugin only)

启用小部件。

- 该方法不接受任何参数。

Code examples:

当点击小部件的任意锚点时启用小部件。

```
this._on( this.element, {
  "click a": function( event ) {
    event.preventDefault();
    this.enable();
  }
});
```

option(optionName)Returns: Object

获取当前与指定的 `optionName` 关联的值。

- **optionName**Type: String要获取的选项的名称。

Code examples:

获得 `width` 选项的值。

```
this.option( "width" );
```

option()Returns: PlainObject

获取一个包含键/值对的对象，键/值对表示当前小部件选项哈希。

- 该方法不接受任何参数。

Code examples:

Log the key and value of each of the widget's options for debugging.

```
var options = this.option();
for ( var key in options ) {
  console.log( key, options[ key ] );
}
```

option(optionName, value)Returns: jQuery (plugin only)

设置与指定的 `optionName` 关联的小部件选项的值。

- **optionName**Type: String要设置的选项的名称。
- **value**Type: Object要为选项设置的值。

Code examples:

设置 `width` 选项为 `500` 。

```
this.option( "width", 500 );
```

option(options)Returns: jQuery (plugin only)

为小部件设置一个或多个选项。

- **options**Type: Object要设置的 option-value 对。

Code examples:

设置 `height` 和 `width` 选项为 `500` 。

```
this.option({
  width: 500,
  height: 500
});
```

widget()Returns: jQuery

返回一个包含原始元素或其他相关的生成元素的 `jQuery` 对象。

- 该方法不接受任何参数。

Code examples:

当创建小部件时，在小部件的原始元素周围放置一个红色的边框。

```
_create: function() {
  this.widget().css( "border", "2px solid red" );
}
```

Events

create(event, ui)Type: `widgetcreate`

当小部件被创建时触发。

- **event**Type: Event
- **ui**Type: Object

Note: `ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

Code examples:

初始化带有指定 create 回调的小部件：

```
$( ".selector" ).widget({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 widgetcreate 事件：

```
$( ".selector" ).on( "widgetcreate", function( event, ui ) {} );
```

Widget Plugin Bridge

Categories: Utilities | Widgets

jQuery.widget.bridge(name, constructor)

Description: `jQuery.widget.bridge()` 方法是 jQuery 部件库（Widget Factory） 的一部分。它扮演着由 `$.widget()` 创建的对象和 jQuery API 之间的中介。

- **jQuery.widget.bridge(name, constructor)**
  - **name**Type: String要创建的插件名称。
  - **constructor**Type: Function()当插件被调用时要实例化的对象。

`$.widget.bridge()` 做如下事情：

- 连接一个常规的 JavaScript 构造函数到 jQuery API。
- 自动创建对象实例，并存储在元素的 `$.data` 缓存内。
- 允许调用公有方法。
- 防止调用私有方法。
- 防止在未初始化的对象上调用方法。
- 防止多个初始化。

jQuery UI 小部件使用 `$.widget( "foo.bar", {} );` 语法定义一个对象来创建。给出一个带有五个 `.foo`， `$( ".foo" ).bar();` 的 DOM 结构将创建 "bar" 对象的五个实例。`$.widget.bridge()` 基于 "bar" 对象和一个公共的 API 在库内工作。因此，您可以通过编写 `$( ".foo" ).bar()` 来创建实例，通过编写 `$( ".foo" ).bar( "baz" )` 来调用方法。

如果您只想一次性初始化并调用方法，那么您所传递给 `jQuery.widget.bridge()` 的对象可以很小：

```
var Highlighter = function( options, element ) {
  this.options = options;
  this.element = $( element );
  this._set( 800 );
};
Highlighter.prototype = {
  toggle: function() {
    this._set( this.element.css( "font-weight") === 400 ? 800 : 400 );
  },
  _set: function(value) {
    this.element.css( "font-weight", value );
  }
};
```

在这里，您需要的只是一个构造函数，接收两个参数：

- `options` ：一个配置选项的对象
- `element` ：该实例在其上创建的 DOM 元素

然后您可以使用桥（bridge）把该对象作为一个 jQuery 插件，且可以在任意的 jQuery 对象上使用它：

```
// Hook up the plugin
$.widget.bridge( "colorToggle", Highlighter );

// Initialize it on divs
$( "div" ).colorToggle().click(function() {
  // Call the public method on click
  $( this ).colorToggle( "toggle" );
});
```

为了使用桥（bridge）的所有特性，您的对象原型需要有一个 `_init()` 方法。该方法在调用插件且实例已存在时调用。在这种情况下，您还需要有一个 `option()` 方法。该方法将会以选项作为第一个参数被调用。如果没有选项，则参数为一个空对象。如需了解 `option` 方法的使用，请查看 `$.Widget` 。

桥（bridge）有一个可选的属性，如果存在：如果对象原型有一个 `widgetFullName` 属性，则该属性将被作为存储和检索实例的键。否则，将使用 name 参数。

Widgets

小部件（Widgets）是功能丰富、有状态的插件，它有一个完整的生命周期，带有方法和事件。您可以查看 部件库（Widget Factory）文档 了解更多详情。

Accordion Widget

把一对标题和内容面板转换成折叠面板。

Autocomplete Widget

自动完成功能根据用户输入值进行搜索和过滤，让用户快速找到并从预设值列表中选择。

Button Widget

可主题化的按钮和按钮集合。

Datepicker Widget

从弹出框或在线日历选择一个日期。

Dialog Widget

在一个交互覆盖层中打开内容。

Also in: Utilities

Widget Factory

使用与所有 jQuery UI 小部件相同的抽象化来创建有状态的 jQuery 插件。

Also in: Utilities

Widget Plugin Bridge

jQuery.widget.bridge() 方法是 jQuery 部件库（Widget Factory）的一部分。它扮演着由 $.widget() 创建的对象和 jQuery API 之间的中介。

Menu Widget

带有鼠标和键盘交互的用于导航的可主题化菜单。

Progressbar Widget

显示一个确定的或不确定的进程状态。

Slider Widget

拖动手柄可以选择一个数值。

Spinner Widget

通过向上/向下按钮和箭头键处理，为输入数值增强文本输入功能。

Tabs Widget

一种多面板的单内容区，每个面板与列表中的标题相关。

Tooltip Widget

可自定义的、可主题化的工具提示框，替代原生的工具提示框。

Accordion Widget

Categories: Widgets

version added: 1.0

Description: 使一对标题和内容面板转换成折叠面板（Accordion）。

QuickNavExamples

Options

- active
- animate
- collapsible
- disabled
- event
- header
- heightStyle
- icons

Methods

- destroy
- disable
- enable
- option
- refresh
- widget

Events

- activate
- beforeActivate
- create

你的accordion容器需要按照一个元素成组的满足拥有配对的头部和内容面板的格式要求：

```
<div id="accordion">
  <h3>First header</h3>
  <div>First content panel</div>
  <h3>Second header</h3>
  <div>Second content panel</div>
</div>
```

Accordions （折叠面板）支持任意的标记，但每个内容面板必须始终是 其相关联头部之后的下一个兄弟节点。请参阅有关如何使用自定义标记结构的 `header` 选项。

面板可以通过设置 `active` 选项来激活。

键盘交互(Keyboard interaction)

当焦点在标题（header）上时，下面的键盘命令可用：

- UP/LEFT - 移动焦点到上一个标题（header）。如果在第一个标题（header）上，则移动焦点到最后一个标题（header）上。
- DOWN/RIGHT - 移动焦点到下一个标题（header）。如果在最后一个标题（header）上，则移动焦点到第一个标题（header）上。
- HOME - 移动焦点到第一个标题（header）上。
- END - 移动焦点到最后一个标题（header）上。
- SPACE/ENTER - 激活与获得焦点的标题（header）相关的面板（panel）。

当焦点在面板（panel）中时，下面的键盘命令可用：

- CTRL+UP: 移动焦点到相关的标题（header）。

主题(Theming)

折叠面板部件（Accordion Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用折叠面板指定的样式，则可以使用下面的 CSS class 名称： ui-accordion：折叠面板的外层容器。 ui-accordion-header：折叠面板的标题。如果标题包含 icons，则标题会另外有个 ui-accordion-icons class。 ui-accordion-content：折叠面板的内容面板。

折叠面板部件（Accordion Widget）使用 jQuery UI CSS framework 框架 来定义它的外观和感观的样式。如果需要使用折叠面板指定的样式，则可以使用下面的 CSS class 名称：

- `ui-accordion`：折叠面板的外层容器。
    - `ui-accordion-header`：折叠面板的标题。如果标题包含 `icons`，则标题会另外有个 `ui-accordion-icons` class。
    - `ui-accordion-content`：折叠面板的内容面板。

依赖

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 特效核心（Effects Core）　（可选的；当与 `animate` 选项一起使用时）

其他注意事项:

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

选项

activeType: Boolean or Integer

Default: `0` 当前打开哪一个面板。支持多个类型：

- **Boolean**:设置 `active` 为 `false` 将折叠所有的面板。这要求 `collapsible` 选项必须为 `true` 。
- **Integer**: 激活打开的面板索引，以零为基础。负值则表示从最后一个面板后退选择面板。

Code examples:

初始化带有指定 `active` 选项的 Accordion（折叠面板）：

```
$( ".selector" ).accordion({ active: 2 });
```

在初始化后，获取或设置 `active` 选项：

```
// getter
var active = $( ".selector" ).accordion( "option", "active" );

// setter
$( ".selector" ).accordion( "option", "active", 2 );
```

animateType: Boolean or Number or String or Object

Default: `{}` 是否使用动画改变面板，且如何使用动画改变面板。支持多个类型：

- **Boolean**: `false` 值将禁用动画。
- **Number**: 默认 easing 动画的持续时间，以毫秒为单位。
- **String**: 默认的持续时间要使用的 easing 动画形式 名称。
- **Object**: `easing` 和 `duration` 属性的动画设置。
  - 上面任意的选项都可以包含 `down` 属性。
  - 当被激活的面板有一个比当前激活面板较低的指数时，发生 "Down" 动画。

Code examples:

初始化带有指定 `animate` 选项的 Accordion（折叠面板）：

```
$( ".selector" ).accordion({ animate: "bounceslide" });
```

在初始化后，获取或设置 `animate` 选项：

```
// getter
var animate = $( ".selector" ).accordion( "option", "animate" );

// setter
$( ".selector" ).accordion( "option", "animate", "bounceslide" );
```

collapsibleType: Boolean

Default: `false` 所有部分是否都可以马上关闭。允许折叠激活的部分。**Code examples:**

初始化带有指定 `collapsible` 选项的 Accordion（折叠面板）：

```
$( ".selector" ).accordion({ collapsible: true });
```

在初始化后，获取或设置 `collapsible` 选项：

```
// getter
var collapsible = $( ".selector" ).accordion( "option", "collapsible" );

// setter
$( ".selector" ).accordion( "option", "collapsible", true );
```

disabledType: Boolean

Default: `false` 如果设置为 `true`，则禁用该 Accordion（折叠面板）。**Code examples:**

初始化带有指定 `disabled` 选项的 Accordion（折叠面板）：

```
$( ".selector" ).accordion({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).accordion( "option", "disabled" );

// setter
$( ".selector" ).accordion( "option", "disabled", true );
```

eventType: String

Default: `"click"` Accordion（折叠面板） 头部会作出反应的事件，用以激活相关的面板。可以指定多个事件，用空格隔开。**Code examples:**

初始化带有指定 `event` 选项的 Accordion（折叠面板） ：

```
$( ".selector" ).accordion({ event: "mouseover" });
```

在初始化后，获取或设置 `event` 选项 ：

```
// getter
var event = $( ".selector" ).accordion( "option", "event" );

// setter
$( ".selector" ).accordion( "option", "event", "mouseover" );
```

headerType: Selector

Default: `"> li > :first-child,> :not(li):even"`

标题元素的选择器，通过主要 accordion 元素上的 `.find()` 进行应用。内容面板必须是紧跟在与其相关的标题后的同级元素。

Code examples:

初始化带有指定 `header` 选项的 Accordion（折叠面板） ：

```
$( ".selector" ).accordion({ header: "h3" });
```

在初始化后，获取或设置 `header` 选项 ：

```
// getter
var header = $( ".selector" ).accordion( "option", "header" );

// setter
$( ".selector" ).accordion( "option", "header", "h3" );
```

heightStyleType: String

Default: `"auto"`

控制 Accordion（折叠面板） 和每个面板的高度。可能的值 ：

- `"auto"` : 所有的面板将会被设置为最高的面板的高度。
- `"fill"` : 基于 accordion 的父元素的高度，扩展到可用的高度。
- `"content"` : 每个面板的高度取决于它的内容。

Code examples:

初始化带有指定 `heightStyle` 选项的 Accordion（折叠面板）：

```
$( ".selector" ).accordion({ heightStyle: "fill" });
```

在初始化后，获取或设置 `heightStyle` 选项：

```
// getter
var heightStyle = $( ".selector" ).accordion( "option", "heightStyle" );

// setter
$( ".selector" ).accordion( "option", "heightStyle", "fill" );
```

iconsType: Object

Default: `{ "header": "ui-icon-triangle-1-e", "activeHeader": "ui-icon-triangle-1-s" }`

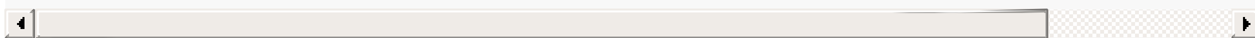
标题要使用的图标，与 jQuery UI CSS 框架提供的图标（Icons） 匹配。设置为 false 则不显示图标。

- header (string, 默认值： "ui-icon-triangle-1-e")
- activeHeader (string, 默认值： "ui-icon-triangle-1-s")

Code examples:

初始化带有指定 `icons` 选项的 Accordion（折叠面板）：

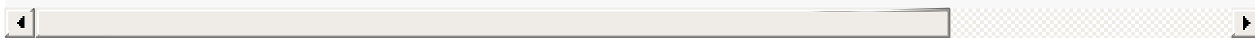
```
$( ".selector" ).accordion({ icons: { "header": "ui-icon-plus", "activeHeader": "ui-icon-
```



在初始化后，获取或设置 `icons` 选项：

```
// getter
var icons = $( ".selector" ).accordion( "option", "icons" );

// setter
$( ".selector" ).accordion( "option", "icons", { "header": "ui-icon-plus", "activeHeader"
```



Methods

destroy()Returns: jQuery (plugin only)

完全移除 accordion 功能。这会把元素返回到它的预初始化状态。

- 该方法不接受任何参数。

Code examples:

调用 destroy 方法：

```
$( ".selector" ).accordion( "destroy" );
```

disable()Returns: jQuery (plugin only)

禁用 accordion。

- 该方法不接受任何参数。

Code examples:

调用 disable 方法：

```
$( ".selector" ).accordion( "disable" );
```

enable()Returns: jQuery (plugin only)

启用 accordion。

- 该方法不接受任何参数。

Code examples:

调用 enable 方法：

```
$( ".selector" ).accordion( "enable" );
```

option(optionName)Returns: Object

获取当前与指定的 `optionName` 关联的值。

- **optionName**Type: String要获取值的选项的名称。

Code examples:

调用该方法：

```
var isDisabled = $( ".selector" ).accordion( "option", "disabled" );
```

option()Returns: PlainObject

获取一个包含键/值对的对象，键/值对表示当前 accordion 选项哈希。

- 该方法不接受任何参数。

Code examples:

调用该方法：

```
var options = $( ".selector" ).accordion( "option" );
```

option(optionName, value)Returns: jQuery (plugin only)

设置与指定的 `optionName` 关联的 accordion 选项的值。

- **optionName**Type: String要设置的选项的名称。
- **value**Type: Object要为选项设置的值。

Code examples:

调用该方法：

```
$( ".selector" ).accordion( "option", "disabled", true );
```

option(options)Returns: jQuery (plugin only)

为 accordion 设置一个或多个选项。

- **options**Type: Object要设置的 option-value 对。

Code examples:

调用该方法：

```
$( ".selector" ).accordion( "option", { disabled: true } );
```

refresh()Returns: jQuery (plugin only)

处理任何在 DOM 中直接添加或移除的标题和面板，并重新计算 accordion 的高度。结果取决于内容和 `heightStyle` 选项。

- 该方法不接受任何参数。

Code examples:

调用 refresh 方法：

```
$( ".selector" ).accordion( "refresh" );
```

widget()Returns: jQuery

返回一个包含 accordion 的 `jQuery` 对象。

- 该方法不接受任何参数。

Code examples:

调用 widget 方法：

```
var widget = $( ".selector" ).accordion( "widget" );
```

Events

activate(event, ui)Type: `accordionactivate`

面板被激活后触发（在动画完成之后）。如果 accordion 之前是折叠的，则 `ui.oldHeader` 和 `ui.oldPanel` 将是空的 jQuery 对象。如果 accordion 正在折叠，则 `ui.newHeader` 和 `ui.newPanel` 将是空的 jQuery 对象。

注意： 由于 `activate` 事件只有在面板激活时才能触发，当创建 accordion 部件时，最初的面板不会触发该事件。如果您需要一个用于部件创建的钩，请使用 `create` 事件。

- **event**Type: Event
- **ui**Type: Object
  - **newHeader**Type: jQuery刚被激活的标题。
  - **oldHeader**Type: jQuery刚被取消激活的标题。
  - **newPanel**Type: jQuery刚被激活的面板。
  - **oldPanel**Type: jQuery刚被取消激活的面板。

Code examples:

初始化带有指定 activate 回调的 accordion：

```
$( ".selector" ).accordion({
  activate: function( event, ui ) {}
});
```

绑定一个事件监听器到 accordionactivate 事件:

```
$( ".selector" ).on( "accordionactivate", function( event, ui ) {} );
```

beforeActivate(event, ui)Type: `accordionbeforeactivate`

面板被激活前直接触发。可以取消以防止面板被激活。如果 accordion 当前是折叠的，则 `ui.oldHeader` 和 `ui.oldPanel` 将是空的 jQuery 对象。如果 accordion 正在折叠，则 `ui.newHeader` 和 `ui.newPanel` 将是空的 jQuery 对象。

- **event**Type: Event
- **ui**Type: Object
    - **newHeader**Type: jQuery将被激活的标题。
    - **oldHeader**Type: jQuery将被取消激活的标题。
    - **newPanel**Type: jQuery将被激活的面板。
    - **oldPanel**Type: jQuery将被取消激活的面板。

Code examples:

初始化带有指定 beforeActivate 回调的 accordion：

```
$( ".selector" ).accordion({
  beforeActivate: function( event, ui ) {}
});
```

绑定一个事件监听器到 accordionbeforeactivate 事件:

```
$( ".selector" ).on( "accordionbeforeactivate", function( event, ui ) {} );
```

create(event, ui)Type: `accordioncreate`

当创建 accordion 时触发。如果 accordion 是折叠的，`ui.header` 和 `ui.panel` 将是空的 jQuery 对象。

- **event**Type: Event
- **ui**Type: Object
    - **header**Type: jQuery激活的标题。
    - **panel**Type: jQuery激活的面板。

Code examples:

初始化带有指定 create 回调的 accordion：

```
$( ".selector" ).accordion({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 accordioncreate 事件:

```
$( ".selector" ).on( "accordioncreate", function( event, ui ) {} );
```

Example:

一个简单的 jQuery UI 折叠面板（Accordion）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>accordion demo</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-ui.cs
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="accordion">
  <h3>Section 1</h3>
  <div>
    <p>Mauris mauris ante, blandit et, ultrices a, suscipit eget.
    Integer ut neque. Vivamus nisi metus, molestie vel, gravida in,
    condimentum sit amet, nunc. Nam a nibh. Donec suscipit eros.
    Nam mi. Proin viverra leo ut odio.</p>
  </div>
  <h3>Section 2</h3>
  <div>
    <p>Sed non urna. Phasellus eu ligula. Vestibulum sit amet purus.
    Vivamus hendrerit, dolor aliquet laoreet, mauris turpis velit,
    faucibus interdum tellus libero ac justo.</p>
  </div>
  <h3>Section 3</h3>
  <div>
    <p>Nam enim risus, molestie et, porta ac, aliquam ac, risus.
    Quisque lobortis.Phasellus pellentesque purus in massa.</p>
    <ul>
      <li>List item one</li>
      <li>List item two</li>
      <li>List item three</li>
    </ul>
  </div>
</div>

<script>
$( "#accordion" ).accordion();
</script>

</body>
</html>
```

Autocomplete Widget

Categories: Widgets

version added: 1.8

Description: 自动完成功能根据用户输入值进行搜索和过滤，让用户快速找到并从预设值列表中选择。

QuickNavExamples

Options

- appendTo
- autoFocus
- delay
- disabled
- minLength
- position
- source

Methods

- close
- destroy
- disable
- enable
- option
- search
- widget

Extension Points

- _renderItem
- _renderMenu
- _resizeMenu

Events

- change
- close
- create
- focus
- open
- response
- search
- select

任何可以接收输入的字段都可以转换为 Autocomplete，即，<input> 元素，<textarea> 元素及带有 `contenteditable` 属性的元素。

通过给 Autocomplete 字段焦点或者在其中输入字符，插件开始搜索匹配的条目并显示供选择的值的列表。通过输入更多的字符，用户可以过滤列表以获得更好的匹配。

该部件可用于选择先前选定的值，比如输入文章标签或者输入从地址簿中输入地址邮件地址。Autocomplete 也可以用来填充相关的信息，比如输入一个城市的名称来获得该城市的邮政编码。

您可以从本地源或者远程源获取数据：本地源适用于小数据集，例如，带有 50 个条目的地址簿；远程源适用于大数据集，比如，带有数百个或者成千上万个条目的数据库。如需了解更多有关自定义数据源的信息，请查看 `source` 选项的文档。

键盘交互(Keyboard interaction)

当菜单打开时，下面的键盘命令可用：

- UP - 移动焦点到上一个项目。如果在第一个项目上，则移动焦点到输入（input）。如果在输入（input）上，则移动焦点到最后一个项目。
- DOWN - 移动焦点到下一个项目。如果在最后一个项目上，则移动焦点到输入（input）。如果在输入（input）上，则移动焦点到第一个项目。
- ESCAPE - 关闭菜单。
- ENTER - 选择当前获得焦点的项目，并关闭菜单。
- TAB - 选择当前获得焦点的项目，关闭菜单，并移动焦点到下一个可聚焦（focusable）的元素。
- PAGE UP/DOWN - 滚动一屏的项目（基于菜单的高度）。

当菜单关闭时，下面的键盘命令可用：

- UP/DOWN - 如果满足 `minLength`，则打开菜单。

主题(Theming)

自动完成部件（Autocomplete Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用自动完成部件指定的样式，则可以使用下面的 CSS class 名称：

- `ui-autocomplete`：用于显示匹配用户的 菜单（menu）
- `ui-autocomplete-input`：自动完成部件（Autocomplete Widget）实例化的 input 元素。

依赖(Dependencies)

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 定位（Position）
- 菜单部件（Menu Widget）

其他注意事项:

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。
- 该部件以编程方式操作元素的值，因此当元素的值改变时不会触发原生的 `change` 事件。

选项

appendToType: Selector

Default: `null`

菜单应该被附加到哪一个元素。当该值为 `null` 时，输入域的父元素将检查 `ui-front` class。如果找到带有 `ui-front` class 的元素，菜单将被附加到该元素。如果未找到带有 `ui-front` class 的元素，不管值为多少，菜单将被附加到 body。

注意: 当建议菜单打开时，`appendTo` 选项不应改变。**Code examples:**

初始化带有指定 `appendTo` 选项的 autocomplete：

```
$( ".selector" ).autocomplete({ appendTo: "#someElem" });
```

在初始化后，获取或设置 `appendTo` 选项：

```
// getter
var appendTo = $( ".selector" ).autocomplete( "option", "appendTo" );

// setter
$( ".selector" ).autocomplete( "option", "appendTo", "#someElem" );
```

autoFocusType: Boolean

Default: `false` 如果设置为 `true` ，当菜单显示时，第一个条目将自动获得焦点。**Code examples:**

初始化带有指定 `autoFocus` 选项的 autocomplete：

```
$( ".selector" ).autocomplete({ autoFocus: true });
```

在初始化后，获取或设置 `autoFocus` 选项：

```
// getter
var autoFocus = $( ".selector" ).autocomplete( "option", "autoFocus" );

// setter
$( ".selector" ).autocomplete( "option", "autoFocus", true );
```

delayType: Integer

Default: `300` 按键和执行搜索之间的延迟，以毫秒计。对于本地数据，采用零延迟是有意义的（更具响应性），但对于远程数据会产生大量的负荷，同时降低了响应性。**Code examples:**

初始化带有指定 `delay` 选项的 autocomplete：

```
$( ".selector" ).autocomplete({ delay: 500 });
```

在初始化后，获取或设置 `delay` 选项：

```
// getter
var delay = $( ".selector" ).autocomplete( "option", "delay" );

// setter
$( ".selector" ).autocomplete( "option", "delay", 500 );
```

disabledType: Boolean

Default: `false` 如果设置为 `true` ，则禁用该 autocomplete。**Code examples:**

初始化带有指定 `disabled` 选项的 autocomplete：

```
$( ".selector" ).autocomplete({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).autocomplete( "option", "disabled" );

// setter
$( ".selector" ).autocomplete( "option", "disabled", true );
```

minLengthType: Integer

Default: `1` 执行搜索前用户必须输入的最小字符数。对于仅带有几项条目的本地数据，通常设置为零，但是当单个字符搜索会匹配几千项条目时，设置个高数值是很有必要的。**Code examples:**

初始化带有指定 `minLength` 选项的 autocomplete：

```
$( ".selector" ).autocomplete({ minLength: 0 });
```

在初始化后，获取或设置 `minLength` 选项：

```
// getter
var minLength = $( ".selector" ).autocomplete( "option", "minLength" );

// setter
$( ".selector" ).autocomplete( "option", "minLength", 0 );
```

positionType: Object

Default: `{ my: "left top", at: "left bottom", collision: "none" }` 标识建议菜单的位置与相关的 input 元素有关系。`of` 选项默认为 input 元素，但是您可以指定另一个定位元素。如需了解各种选项的更多细节，请查看 jQuery UI Position。**Code examples:**

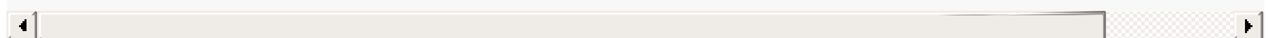
初始化带有指定 `position` 选项的 autocomplete：

```
$( ".selector" ).autocomplete({ position: { my : "right top", at: "right bottom" } });
```

在初始化后，获取或设置 `position` 选项：

```
// getter
var position = $( ".selector" ).autocomplete( "option", "position" );

// setter
$( ".selector" ).autocomplete( "option", "position", { my : "right top", at: "right botto
```



sourceType: Array or String or Function(Object request, Function response(Object data))

Default: `none`; 必须指定定义要使用的数据，必须指定。

独立于您要使用的变量，标签总是被视为文本。如果您想要标签被视为 html，您可以使用 Scott González' html 扩展 。演示侧重于 `source` 选项的不同变量 - 您可以查找其中匹配您的使用情况的那个，并查看相关代码。

支持多个类型：

- **Array**：可用于本地数据的一个数组。支持两种格式：
    - 字符串数组：`[ "Choice1", "Choice2" ]`
    - 带有 `label` 和 `value` 属性的对象数组：`[ { label: "Choice1", value: "value1" }, ... ]` label 属性显示在建议菜单中。当用户选择一个条目时，value 将被插入到 input 元素中。如果只是指定了一个属性，则该属性将被视为 label 和 value，例如，如果您只提供了 `value` 属性，value 也会被视为标签。
- **String**：当使用一个字符串，Autocomplete 插件希望该字符串指向一个能返回 JSON 数据的 URL 资源。它可以是在相同的主机上，也可以是在不同的主机上（必须提供 JSONP）。Autocomplete 插件不过滤结果，而是通过一个 `term` 字段添加了一个查询字符串，用于服务器端脚本过滤结果。例如，如果 `source` 选项设置为 `"http://example.com"`，且用户键入了 `foo`，GET 请求则为 `http://example.com?term=foo`。数据本身的格式可以与前面描述的本地数据的格式相同。
- **Function**：第三个变量，一个回调函数，提供最大的灵活性，可用于连接任何数据源到 Autocomplete。回调函数接受两个参数：
    - 一个 `request` 对象，带有一个 `term` 属性，表示当前文本输入中的值。例如，如果用户在 city 字段输入 `"new yo"`，则 Autocomplete term 等同于 `"new yo"`。
    - 一个 `response` 回调函数，提供单个参数：建议给用户的数据。该数据应基于被提供的 term 进行过滤，且可以是上面描述的本地数据的任何格式。用于在请求期间提供自定义 source 回调来处理错误。即使遇到错误，您也必须调用 `response` 回调函数。这就确保了小部件总是正确的状态。

    当过滤本地数据时，您可以使用内置的 `$.ui.autocomplete.escapeRegex` 函数。它会接受一个字符串参数，转义所有的正则表达式字符，让结果安全地传递到 `new RegExp()`。

Code examples:

初始化带有指定 `source` 选项的 autocomplete：

```
$( ".selector" ).autocomplete({ source: [ "c++", "java", "php", "coldfusion", "javascript
```

在初始化后，获取或设置 `source` 选项：

```
// getter
var source = $( ".selector" ).autocomplete( "option", "source" );

// setter
$( ".selector" ).autocomplete( "option", "source", [ "c++", "java", "php", "coldfusion",
```

Methods

close()Returns: jQuery (plugin only)

关闭 Autocomplete 菜单。当与 `search` 方法结合使用时，可用于关闭打开的菜单。

- 该方法不接受任何参数。

Code examples:

调用 close 方法：

```
$( ".selector" ).autocomplete( "close" );
```

destroy()Returns: jQuery (plugin only)

完全移除 autocomplete 功能。这会把元素返回到它的预初始化状态。

- 该方法不接受任何参数。

Code examples:

调用 destroy 方法：

```
$( ".selector" ).autocomplete( "destroy" );
```

disable()Returns: jQuery (plugin only)

禁用 autocomplete。

- 该方法不接受任何参数。

Code examples:

调用 disable 方法：

```
$( ".selector" ).autocomplete( "disable" );
```

enable()Returns: jQuery (plugin only)

启用 autocomplete。

- 该方法不接受任何参数。

Code examples:

调用 enable 方法：

```
$( ".selector" ).autocomplete( "enable" );
```

option(optionName)Returns: Object

获取当前与指定的 `optionName` 关联的值。

- **optionName**Type: String要获取值的选项的名称。

Code examples:

调用该方法：

```
var isDisabled = $( ".selector" ).autocomplete( "option", "disabled" );
```

option()Returns: PlainObject

获取一个包含键/值对的对象，键/值对表示当前 autocomplete 选项哈希。

- 该方法不接受任何参数。

Code examples:

调用该方法：

```
var options = $( ".selector" ).autocomplete( "option" );
```

option(optionName, value)Returns: jQuery (plugin only)

设置与指定的 `optionName` 关联的 autocomplete 选项的值。

- **optionName**Type: String要设置的选项的名称。
- **value**Type: Object要为选项设置的值。

Code examples:

调用该方法：

```
$( ".selector" ).autocomplete( "option", "disabled", true );
```

option(options)Returns: jQuery (plugin only)

为 autocomplete 设置一个或多个选项。

- **options**Type: Object要设置的 option-value 对。

Code examples:

调用该方法：

```
$( ".selector" ).autocomplete( "option", { disabled: true } );
```

search([value])Returns: jQuery (plugin only)

触发 `search` 事件，如果该事件未被取消则调用数据源。当被点击时，可被类似选择框按钮用来打开建议。当不带参数调用该方法时，则使用当前输入的值。可带一个空字符串和 `minLength: 0` 进行调用，来显示所有的条目。

- **value**Type: String

Code examples:

调用 search 方法：

```
$( ".selector" ).autocomplete( "search", "" );
```

widget()Returns: jQuery

返回一个包含菜单元素的 `jQuery` 对象。虽然菜单项不断地被创建和销毁。菜单元素本身会在初始化时创建，并不断的重复使用。

- 该方法不接受任何参数。

Code examples:

调用 widget 方法：

```
$( ".selector" ).autocomplete( "widget" );
```

Extension Points

自动完成部件（Autocomplete Widget）通过 部件库（Widget Factory） 创建的，且可被扩展。当对部件进行扩展时，您可以重载或者添加扩展部件的行为。下面提供的方法作为扩展点，与上面列出的 插件方法 具有相同的 API 稳定性。如需了解更多有关小部件扩展的知识，请查看 通过部件库（Widget Factory）扩展小部件。

_renderItem(ul, item)Returns: jQuery

控制在部件菜单中每个选项的创建的格式化方法 该方法必须创建一个新的 `<li>` 元素，追加到菜单中，并返回它。

注意: 为了与*menu*小部件兼容，这时创建的 *`<ul>`* 元素必须包含一个 *`<a>`* 元素。请看下面的例子。

- **ul**Type: jQuery新创建的 `<li>` 元素必须追加到的 `<ul>` 元素。
- **item**Type: Object
  - **label**Type: String条目显示的字符串。
  - **value**Type: String当条目被选中时插入到输入框中的值。

Code examples:

添加条目的值作为 `<li>` 上的 data 属性。

```
_renderItem: function( ul, item ) {
  return $( "<li>" )
    .attr( "data-value", item.value )
    .append( $( "<a>" ).text( item.label ) )
    .appendTo( ul );
}
```

_renderMenu(ul, items)Returns: jQuery (plugin only)

控制建立部件菜单的方法。该方法传递一个空 `<ul>` 和匹配用户输入的术语的项目数组。 创建当个的 `<li>` 元素应该委派给 `_renderItemData()` 。

- **ul**Type: jQuery一个要作为小部件的菜单使用的空的 `<ul>` 元素。
- **items**Type: Array一个数组，数组元素为匹配用户输入的条目。每个条目是一个带有 `label` 和 `value` 属性的对象。

Code examples:

添加一个 CSS class 名称到旧的菜单项。

```
_renderMenu: function( ul, items ) {
  var that = this;
  $.each( items, function( index, item ) {
    that._renderItemData( ul, item );
  });
  $( ul ).find( "li:odd" ).addClass( "odd" );
}
```

thod-_resizeMenu">

_resizeMenu()Returns: jQuery (plugin only)

该方法负责在菜单显示前调整菜单尺寸。菜单元素可通过 `this.menu.element` 使用。

- 该方法不接受任何参数。

Code examples:

菜单总是显示为 500 像素宽。

```
_resizeMenu: function() {
  this.menu.element.outerWidth( 500 );
}
```

ction id="events">

Events

change(event, ui)Type: `autocompletechange`

如果输入域的值改变则触发该事件。

- **event**Type: Event
- **ui**Type: Object
  - **item**Type: Object从菜单中选择的条目，否则属性为 `null`。

Code examples:

初始化带有指定 change 回调的 autocomplete：

```
$( ".selector" ).autocomplete({
  change: function( event, ui ) {}
});
```

绑定一个事件监听器到 autocompletechange 事件：

```
$( ".selector" ).on( "autocompletechange", function( event, ui ) {} );
```

close(event, ui)Type: `autocompleteclose`

当菜单隐藏时触发。不是每一个 `close` 事件都伴随着 `change` 事件。

- **event**Type: Event
- **ui**Type: Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

Code examples:

初始化带有指定 close 回调的 autocomplete：

```
$( ".selector" ).autocomplete({
  close: function( event, ui ) {}
});
```

绑定一个事件监听器到 autocompleteclose 事件：

```
$( ".selector" ).on( "autocompleteclose", function( event, ui ) {} );
```

create(event, ui)Type: `autocompletecreate`

当创建 autocomplete 时触发。

- **event**Type: Event
- **ui**Type: Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

Code examples:

初始化带有指定 create 回调的 autocomplete：

```
$( ".selector" ).autocomplete({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 autocompletecreate 事件：

```
$( ".selector" ).on( "autocompletecreate", function( event, ui ) {} );
```

focus(event, ui)Type: `autocompletefocus`

当焦点移动到一个条目上（未选择）时触发。默认的动作是把文本域中的值替换为获得焦点的条目的值，即使该事件是通过键盘交互触发的。

取消该事件会阻止值被更新，但不会阻止菜单项获得焦点。

- **event**Type: Event
- **ui**Type: Object
  - **item**Type: Object获得焦点的条目。

Code examples:

初始化带有指定 focus 回调的 autocomplete：

```
$( ".selector" ).autocomplete({
  focus: function( event, ui ) {}
});
```

绑定一个事件监听器到 autocompletefocus 事件：

```
$( ".selector" ).on( "autocompletefocus", function( event, ui ) {} );
```

open(event, ui)Type: `autocompleteopen`

当打开建议菜单或者更新建议菜单时触发。

- **event**Type: Event
- **ui**Type: Object

注意： `ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

Code examples:

初始化带有指定 open 回调的 autocomplete：

```
$( ".selector" ).autocomplete({
  open: function( event, ui ) {}
});
```

绑定一个事件监听器到 autocompleteopen 事件：

```
$( ".selector" ).on( "autocompleteopen", function( event, ui ) {} );
```

response(event, ui)Type: `autocompleteresponse`

在搜索完成后菜单显示前触发。用于建议数据的本地操作，其中自定义的 `source` 选项回调不是必需的。该事件总是在搜索完成时触发，如果搜索无结果或者禁用了 Autocomplete，导致菜单未显示，该事件一样会被触发。

- **event**Type: Event

- **ui**Type: Object
    - **content**Type: Array包含响应数据，且可被修改来改变显示结果。该数据已经标准化，所以如果您要修改数据，请确保每个条目都包含 `value` 和 `label` 属性。

Code examples:

初始化带有指定 response 回调的 autocomplete：

```
$( ".selector" ).autocomplete({
  response: function( event, ui ) {}
});
```

绑定一个事件监听器到 autocompleteresponse 事件：

```
$( ".selector" ).on( "autocompleteresponse", function( event, ui ) {} );
```

search(event, ui)Type: `autocompletesearch`

在搜索执行前满足 `minLength` 和 `delay` 后触发。如果取消该事件，则不会提交请求，也不会提供建议条目。

- **event**Type: Event
- **ui**Type: Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

Code examples:

初始化带有指定 search 回调的 autocomplete：

```
$( ".selector" ).autocomplete({
  search: function( event, ui ) {}
});
```

绑定一个事件监听器到 autocompletesearch 事件：

```
$( ".selector" ).on( "autocompletesearch", function( event, ui ) {} );
```

select(event, ui)Type: `autocompleteselect`

当从菜单中选择条目时触发。默认的动作是把文本域中的值替换为被选中的条目的值。

取消该事件会阻止值被更新，但不会阻止菜单关闭。

- **event**Type: Event
- **ui**Type: Object

- **item**Type: ObjectAn Object with `label` and `value` properties for the selected option.

Code examples:

初始化带有指定 select 回调的 autocomplete：

```
$( ".selector" ).autocomplete({
  select: function( event, ui ) {}
});
```

绑定一个事件监听器到 autocompleteselect 事件：

```
$( ".selector" ).on( "autocompleteselect", function( event, ui ) {} );
```

Examples:

Example: 一个简单的 jQuery UI 自动完成（Autocomplete）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>autocomplete demo</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-ui.cs
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<label for="autocomplete">Select a programming language: </label>
<input id="autocomplete">

<script>
$( "#autocomplete" ).autocomplete({
  source: [ "c++", "java", "php", "coldfusion", "javascript", "asp", "ruby" ]
});
</script>

</body>
</html>
```

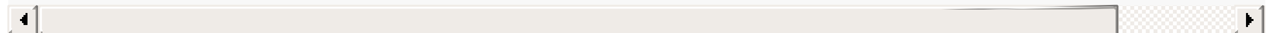
Example: Using a custom source callback to match only the beginning of terms

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>autocomplete demo</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-ui.cs
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<label for="autocomplete">Select a programming language: </label>
<input id="autocomplete">

<script>
var tags = [ "c++", "java", "php", "coldfusion", "javascript", "asp", "ruby" ];
$( "#autocomplete" ).autocomplete({
  source: function( request, response ) {
          var matcher = new RegExp( "^" + $.ui.autocomplete.escapeRegex( request.term ),
          response( $.grep( tags, function( item ){
              return matcher.test( item );
          }) );
      }
});
</script>

</body>
</html>
```



Button Widget

Categories: Widgets

version added: 1.8

Description: 可主题化的按钮和按钮集合。

QuickNavExamples

Options

- disabled
- icons
- label
- text

Methods

- destroy
- disable
- enable
- option
- refresh
- widget

Events

- create

按钮部件（Button Widget）加强了标准表单元素的功能，比如按钮（button）、输入（input）、锚（anchor），用适当的悬停（hover）和激活（active）样式来主题化按钮。

除了基本的按钮，单选按钮和复选框（input 类型为 radio 和 checkbox）也可以转换为按钮。相关的标签（label）设计成按钮的样式，点击时更新底层的输入。为了能正常工作，需要给 input 一个 `id` 属性，并指向标签（label）的 `for` 属性。不要把 input 放在标签（label）内，否则会引起可访问性问题。

为了分组单选按钮，Button 也提供了一个额外的小部件，名为 Buttonset。Buttonset 通过选择一个容器元素（包含单选按钮）并调用 `.buttonset()` 来使用。Buttonset 也提供了可视化分组，因此当有一组按钮时都可考虑使用它。它会选择所有的后代，并对它们应用 `.button()`。您可以启用和禁用一个按钮集，这将会启用和禁用所有包含的按钮。销毁按钮集会调用每个按钮的 `destroy` 方法。对于分组的单选按钮和复选框按钮，推荐使用带有 `legend` 的 `fieldset` 来提供一个可访问的分组标签。

当使用一个类型为 button、submit 或 reset 的 input 时，仅限于支持纯文本无图标标签。

主题

按钮部件（Button Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用按钮指定的样式，则可以使用下面的 CSS class 名称：

- `ui-button`：表示按钮的 DOM 元素。该元素会根据 text 和 icons 选项添加下列 class 之一：`ui-button-text-only`、`ui-button-icon-only`、`ui-button-icons-only`、`ui-button-t`。
    - `ui-button-icon-primary`：用于显示按钮主要图标的元素。只有当主要图标在 icons 选项中提供时才呈现。
    - `ui-button-text`：在按钮的文本内容周围的容器。
    - `ui-button-icon-secondary`：用于显示按钮的次要图标。只有当次要图标在 icons 选项中提供时才呈现。
- `ui-buttonset`：Buttonset 的外层容器。

依赖

- UI 核心（UI Core）
- 部件库（Widget Factory）

其他注意事项：

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

Options

disabledType: Boolean

Default: `false` 如果设置为 `true`，则禁用该 button。**Code examples:**

初始化带有指定 `disabled` 选项的 button：

```
$( ".selector" ).button({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).button( "option", "disabled" );

// setter
$( ".selector" ).button( "option", "disabled", true );
```

iconsType: Object

Default: `{ primary: null, secondary: null }`

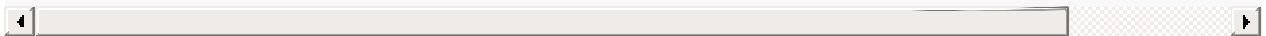
要显示的图标，包括带有文本的图标和不带有文本的图标（查看 `text` 选项）。默认情况下，主图标显示在标签文本的左边，副图标显示在右边。显示位置可通过 CSS 进行控制。

`primary` 和 `secondary` 属性值必须是 图标 class 名称，例如，`"ui-icon-gear"` 。如果只使用一个图标，则 `icons: { primary: "ui-icon-locked" }` 。如果使用两个图标，则 `icons: { primary: "ui-icon-gear", secondary: "ui-icon-triangle-1-s" }` 。

Code examples:

初始化带有指定 `icons` 选项的 button：

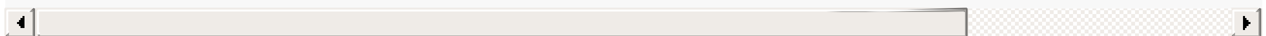
```
$( ".selector" ).button({ icons: { primary: "ui-icon-gear", secondary: "ui-icon-triangle-
```



在初始化后，获取或设置 `icons` 选项：

```
// getter
var icons = $( ".selector" ).button( "option", "icons" );

// setter
$( ".selector" ).button( "option", "icons", { primary: "ui-icon-gear", secondary: "ui-ico
```



labelType: String

Default: `null` 要显示在按钮中的文本。当未指定时（ `null` ），则使用元素的 HTML 内容，或者如果元素是一个 submit 或 reset 类型的 input 元素，则使用它的 `value` 属性，或者如果元素是一个 radio 或 checkbox 类型的 input 元素，则使用相关的 label 元素的 HTML 内容。**Code examples:**

初始化带有指定 `label` 选项的 button：

```
$( ".selector" ).button({ label: "custom label" });
```

在初始化后，获取或设置 `label` 选项：

```
// getter
var label = $( ".selector" ).button( "option", "label" );

// setter
$( ".selector" ).button( "option", "label", "custom label" );
```

textType: Boolean

Default: `true` 是否显示标签。当设置为 `false` 时，不显示文本，但是此时必须启用 `icons` 选项，否则 `text` 选项将被忽略。**Code examples:**

初始化带有指定 `text` 选项的 button：

```
$( ".selector" ).button({ text: false });
```

在初始化后，获取或设置 `text` 选项：

```
// getter
var text = $( ".selector" ).button( "option", "text" );

// setter
$( ".selector" ).button( "option", "text", false );
```

Methods

destroy()Returns: jQuery (plugin only)

完全移除 button 功能。这会把元素返回到它的预初始化状态。

- 该方法不接受任何参数。

Code examples:

调用 destroy 方法：

```
$( ".selector" ).button( "destroy" );
```

disable()Returns: jQuery (plugin only)

禁用 button。

- 该方法不接受任何参数。

Code examples:

调用 disable 方法：

```
$( ".selector" ).button( "disable" );
```

enable()Returns: jQuery (plugin only)

启用 button。

- 该方法不接受任何参数。

Code examples:

调用 enable 方法：

```
$( ".selector" ).button( "enable" );
```

option(optionName)Returns: Object

获取当前与指定的 `optionName` 关联的值。

- **optionName**Type: String要获取值的选项的名称。

Code examples:

调用该方法：

```
var isDisabled = $( ".selector" ).button( "option", "disabled" );
```

option()Returns: PlainObject

获取一个包含键/值对的对象，键/值对表示当前 button 选项哈希。

- 该方法不接受任何参数。

Code examples:

调用该方法：

```
var options = $( ".selector" ).button( "option" );
```

option(optionName, value)Returns: jQuery (plugin only)

设置与指定的 `optionName` 关联的 button 选项的值。

- **optionName**Type: String要设置的选项的名称。
- **value**Type: Object要为选项设置的值。

Code examples:

调用该方法：

```
$( ".selector" ).button( "option", "disabled", true );
```

option(options)Returns: jQuery (plugin only)

为 button 设置一个或多个选项。

- **options**Type: Object要设置的 option-value 对。

Code examples:

调用该方法：

```
$( ".selector" ).button( "option", { disabled: true } );
```

refresh()Returns: jQuery (plugin only)

刷新按钮的视觉状态。用于在以编程方式改变原生元素的选中状态或禁用状态后更新按钮状态。

- 该方法不接受任何参数。

Code examples:

调用 refresh 方法：

```
$( ".selector" ).button( "refresh" );
```

widget()Returns: jQuery

返回一个包含 button 的 `jQuery` 对象。

- 该方法不接受任何参数。

Code examples:

调用 widget 方法：

```
var widget = $( ".selector" ).button( "widget" );
```

Events

create(event, ui)Type: `buttoncreate`

当创建按钮 button 时触发。

- **event**Type: Event
- **ui**Type: Object

注意： `ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

Code examples:

初始化带有指定 create 回调的button

```
$( ".selector" ).button({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 buttoncreate 事件：

```
$( ".selector" ).on( "buttoncreate", function( event, ui ) {} );
```

Examples:

Example: 一个简单的 jQuery UI 按钮（Button）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>button demo</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-ui.cs
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<button>Button label</button>

<script>
$( "button" ).button();
</script>

</body>
</html>
```

Example: 一个简单的 jQuery UI 按钮集（Buttonset）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>button demo</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-ui.cs
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<form>
  <fieldset>
    <legend>Favorite jQuery Project</legend>
    <div id="radio">
      <input type="radio" id="sizzle" name="project">
      <label for="sizzle">Sizzle</label>

      <input type="radio" id="qunit" name="project" checked="checked">
      <label for="qunit">QUnit</label>

      <input type="radio" id="color" name="project">
      <label for="color">Color</label>
    </div>
  </fieldset>
</form>

<script>
$( "#radio" ).buttonset();
</script>

</body>
</html>
```

Datepicker Widget

Categories: Widgets

version added: 1.0

Description: 从弹出框或内联日历选择一个日期。

QuickNavExamples

Options

- altField
- altFormat
- appendText
- autoSize
- beforeShow
- beforeShowDay
- buttonImage
- buttonImageOnly
- buttonText
- calculateWeek
- changeMonth
- changeYear
- closeText
- constrainInput
- currentText
- dateFormat
- dayNames
- dayNamesMin
- dayNamesShort
- defaultDate
- duration
- firstDay
- gotoCurrent
- hideIfNoPrevNext
- isRTL

- maxDate
- minDate
- monthNames
- monthNamesShort
- navigationAsDateFormat
- nextText
- numberOfMonths
- onChangeMonthYear
- onClose
- onSelect
- prevText
- selectOtherMonths
- shortYearCutoff
- showAnim
- showButtonPanel
- showCurrentAtPos
- showMonthAfterYear
- showOn
- showOptions
- showOtherMonths
- showWeek
- stepMonths
- weekHeader
- yearRange
- yearSuffix

Methods

- destroy
- dialog
- getDate
- hide
- isDisabled
- option
- refresh
- setDate
- show
- widget

Events

jQuery UI 日期选择器（Datepicker）是向页面添加日期选择功能的高度可配置插件。您可以自定义日期格式和语言，限制可选择的日期范围，添加按钮和其他导航选项。

默认情况下，当相关的文本域获得焦点时，在一个小的覆盖层打开日期选择器。对于一个内联的日历，只需简单地将日期选择器附加到 div 或者 span 上。

键盘交互

当日期选择器打开时，下面的键盘命令可用：

- PAGE UP：移到上一个月。
- PAGE DOWN：移到下一个月。
- CTRL+PAGE UP：移到上一年。
- CTRL+PAGE DOWN：移到下一年。
- CTRL+HOME：移到当前月份。如果日期选择器是关闭的则打开。
- CTRL+LEFT：移到上一天。
- CTRL+RIGHT：移到下一天。
- CTRL+UP：移到上一周。
- CTRL+DOWN：移到下一周。
- ENTER：选择聚焦的日期。
- CTRL+END：关闭日期选择器，并清除日期。
- ESCAPE：关闭日期选择器，不做任何选择。

实用功能

$.datepicker.setDefaults(settings)

为所有的日期选择器改变默认设置。

使用 `option()` 方法来改变个别实例的设置。

Code examples:

设置所有的日期选择器在获得焦点时或点击图标时打开。

```
$.datepicker.setDefaults({
  showOn: "both",
  buttonImageOnly: true,
  buttonImage: "calendar.gif",
  buttonText: "Calendar"
});
```

设置所有的日期选择器都有法语文本。

```
$.datepicker.setDefaults( $.datepicker.regional[ "fr" ] );
```

$.datepicker.formatDate(format, date, settings)

格式化日期为一个带有指定格式的字符串值。

格式可以是下列组合：

- d - 一月中的第几天（没有前导零）
- dd - 一月中的第几天（两位数）
- o - 一年中的第几天（没有前导零）
- oo - 一年中的第几天（三位数）
- D - 天的短名称
- DD - 天的长名称
- m - 一年中的第几月（没有前导零）
- mm - 一年中的第几月（两位数）
- M - 月的短名称
- MM - 月的长名称
- y - 年（两位数）
- yy - 年（四位数）
- @ - Unix 时间戳（ms since 01/01/1970）
- ! - Windows 钟表（100ns since 01/01/0001）
- '...' - 文本
- '' - 单引号
- 其他 - 文本

也有一些 `$.datepicker` 预定义的标准日期格式：

- ATOM - 'yy-mm-dd' （与 RFC 3339/ISO 8601 相同）
- COOKIE - 'D, dd M yy'
- ISO_8601 - 'yy-mm-dd'
- RFC_822 - 'D, d M y' （参照 RFC 822）
- RFC_850 - 'DD, dd-M-y' （参照 RFC 850）
- RFC_1036 - 'D, d M y' （参照 RFC 1036）
- RFC_1123 - 'D, d M yy' （参照 RFC 1123）
- RFC_2822 - 'D, d M yy' （参照 RFC 2822）
- RSS - 'D, d M y' （与 RFC 822 相同）
- TICKS - '!'
- TIMESTAMP - '@'
- W3C - 'yy-mm-dd' （与 ISO 8601 相同）

Code examples:

以 ISO 格式显示日期。产生 "2007-01-26"。

```
$.datepicker.formatDate( "yy-mm-dd", new Date( 2007, 1 - 1, 26 ) );
```

以扩展法语格式显示日期。产生 "Samedi, Juillet 14, 2007"。

```
$.datepicker.formatDate( "DD, MM d, yy", new Date( 2007, 7 - 1, 14 ), {
  dayNamesShort: $.datepicker.regional[ "fr" ].dayNamesShort,
  dayNames: $.datepicker.regional[ "fr" ].dayNames,
  monthNamesShort: $.datepicker.regional[ "fr" ].monthNamesShort,
  monthNames: $.datepicker.regional[ "fr" ].monthNames
});
```

$.datepicker.parseDate(format, value, settings)

从一个指定格式的字符串值中提取日期。

格式可以是下列组合：

- d - 一月中的第几天（没有前导零）
- dd - 一月中的第几天（两位数）
- o - 一年中的第几天（没有前导零）
- oo - 一年中的第几天（三位数）
- D - 星期几的短名称
- DD - 星期几的长名称
- m - 一年中的第几月（没有前导零）
- mm - 一年中的第几月（两位数）
- M - 月的短名称
- MM - 月的长名称
- y - 年（两位数）
- yy - 年（四位数）
- @ - Unix 时间戳（ms since 01/01/1970）
- ! - Windows 钟表（100ns since 01/01/0001）
- '...' - 文本
- '' - 单引号
- 其他 - 文本

一些可能被抛出的异常：

- 'Invalid arguments' - 如果格式或值为空则抛出此异常。
- 'Missing number at position nn' - 如果格式显示一个未找到的数值则抛出此异常。
- 'Unknown name at position nn' - 如果格式显示一个未找到的星期几名称或月份名称则抛出此异常。
- 'Unexpected literal at position nn' - 如果格式显示一个未找到的文本值则抛出此异常。
- 'Invalid date' - 如果日期无效则抛出此异常，比如 '31/02/2007'。

Code examples:

提取一个 ISO 格式的日期。

```
$.datepicker.parseDate( "yy-mm-dd", "2007-01-26" );
```

提取一个扩展法语格式的日期。

```
$.datepicker.parseDate( "DD, MM d, yy", "Samedi, Juillet 14, 2007", {
  shortYearCutoff: 20,
  dayNamesShort: $.datepicker.regional[ "fr" ].dayNamesShort,
  dayNames: $.datepicker.regional[ "fr" ].dayNames,
  monthNamesShort: $.datepicker.regional[ "fr" ].monthNamesShort,
  monthNames: $.datepicker.regional[ "fr" ].monthNames
});
```

$.datepicker.iso8601Week(date)

确定一个给定的日期在一年中的第几周：1 到 53。

该函数使用 ISO 8601 定义一周：一周从星期一开始，每一年的第一周包含 1 月 4 日。这意味着上一年至多有三天可能包含在当年的第一周中，当年至多有三天可能包含在上一年的最后一周中。

该函数是 `calculateWeek` 选项的默认实现。

Code examples:

查找日期在一年中的第几周。

```
$.datepicker.iso8601Week( new Date( 2007, 1 - 1, 26 ) );
```

$.datepicker.noWeekends

设置如 beforeShowDay 函数，防止选择周末。

我们可以在 `beforeShowDay` 选项中提供 `noWeekends()` 函数，用来计算所有工作日，提供一个 `true` / `false` 值的数组，用来指示日期是否可选择。

Code examples:

设置 DatePicker，让周末不可选。

```
$( "#datepicker" ).datepicker({
  beforeShowDay: $.datepicker.noWeekends
});
```

局限

日期选择器提供了迎合不同的语言和日期格式本地化内容的支持。每个本地化包含在名称后追加语言代码的文件中，例如法语为 `jquery.ui.datepicker-fr.js` 。所需的本地化文件需要包含在主要的日期选择器代码后面。每个本地化文件添加了它自己的设置到可用的本地化集合中，所有实例自动应用这些设置为默认设置。

`$.datepicker.regional` 属性保存了一个本地化数组，以语言代码作为前置，默认前置为 `""` ，表示英语。每个条目是一个带有下列属性的对象：`closeText` 、 `prevText` 、 `nextText` 、 `currentText` 、 `monthNames` 、 `monthNamesShort` 、 `dayNames` 、 `dayNamesShort` 、 `dayNamesMin` 、 `weekHeader` 、 `dateFormat` 、 `firstDay` 、 `isRTL` 、 `showMonthAfterYear` 和 `yearSuffix` 。

您可以通过下面代码恢复默认的本地化：

```
$.datepicker.setDefaults( $.datepicker.regional[ "" ] );
```

您可以通过下面代码覆盖一个特定地点的日期选择器：

```
$( selector ).datepicker( $.datepicker.regional[ "fr" ] );
```

主题

日期选择器部件（Datepicker Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用日期选择器指定的样式，则可以使用下面的 CSS class 名称：

- `ui-datepicker` ：日期选择器的外层容器。如果日期选择器是内联的，该元素会另外带有一个 `ui-datepicker-inline` class。如果设置了 `isRTL` 选项，该元素会另外带有一个 `ui-datepicker-rtl` class。
    - `ui-datepicker-header` ：日期选择器的头部容器。
        - `ui-datepicker-prev` ：用于选择上一月的控件。
        - `ui-datepicker-next` ：用于选择下一月的控件。
        - `ui-datepicker-title` ：日期选择器包含月和年的标题容器。
            - `ui-datepicker-month` ：月的文本显示，如果设置了 `changeMonth` 选项则显示 `<select>` 元素。
            - `ui-datepicker-year` ：年的文本显示，如果设置了 `changeYear` 选项则显示 `<select>` 元素。
    - `ui-datepicker-calendar` ：包含日历的表格。
        - `ui-datepicker-week-end` ：周末的单元格。：Cells containing weekend days.
        - `ui-datepicker-other-month` ：发生在某月但不是当前月天数的单元格。
        - `ui-datepicker-unselectable` ：用户不可选择的单元格。
        - `ui-datepicker-current-day` ：已选中日期的单元格。
        - `ui-datepicker-today` ：当天日期的单元格。
    - `ui-datepicker-buttonpane` ：当设置 `showButtonPanel` 选项时使用按钮面板（buttonpane）。
        - `ui-datepicker-current` ：用于选择当天日期的按钮。

如果 `numberOfMonths` 选项用于显示多个月份，则使用一些额外的 class：

- `ui-datepicker-multi`：一个多月份日期选择器的最外层容器。该元素会根据要显示的月份个数另外带有 `ui-datepicker-multi-2`、`ui-datepicker-multi-3` 或 `ui-datepicker-multi-4` class 名称。
    - `ui-datepicker-group`：分组内单独的选择器。该元素会根据它在分组中的位置另外带有 `ui-datepicker-group-first`、`ui-datepicker-group-middle` 或 `ui-datepicker-group-last` class 名称。

依赖

- UI 核心（UI Core）
- 特效核心（Effects Core）（可选的；当与 `showAnim` 选项一起使用时）

其他注意事项：

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。
- 该部件以编程方式操作元素的值，因此当元素的值改变时不会触发原生的 `change` 事件。
- 不支持在 `<input type="date">` 上创建日期选择器，因为会造成与本地选择器的 UI 冲突。

Options

altFieldType: Selector or jQuery or Element

Default: `""` 一个input元素， 使用选择器 选择的另一个地方更新datepicker选择的日期. 使用 `altFormat` 指定的这一区域设置如下改变格式的日期(使用input最直观). 如果没有代替的区域则使用空白.**Code examples:**

初始化带有指定 `altField` 选项的 datepicker：

```
$( ".selector" ).datepicker({ altField: "#actualDate" });
```

在初始化后，获取或设置 `altField` 选项：

```
// getter
var altField = $( ".selector" ).datepicker( "option", "altField" );

// setter
$( ".selector" ).datepicker( "option", "altField", "#actualDate" );
```

altFormatType: String

Default: `""` `dateFormat` 被 `altField` 项所使用。 它的目的是允许选择一种日期格式显示给用户以供选择，而实际的格式则是来自后台。 对于可能的格式的完整列表，请参阅 `formatDate` 函数。**Code examples:**

初始化带有指定 `altFormat` 选项的 datepicker：

```
$( ".selector" ).datepicker({ altFormat: "yy-mm-dd" });
```

在初始化后，获取或设置 `altFormat` 选项：

```
// getter
var altFormat = $( ".selector" ).datepicker( "option", "altFormat" );

// setter
$( ".selector" ).datepicker( "option", "altFormat", "yy-mm-dd" );
```

appendTextType: String

Default: `""` 显示每个日期字段后面的文本， 例如，以显示所需的格式。**Code examples:**

初始化带有指定 `appendText` 选项的 datepicker：

```
$( ".selector" ).datepicker({ appendText: "(yyyy-mm-dd)" });
```

在初始化后，获取或设置 `appendText` 选项：

```
// getter
var appendText = $( ".selector" ).datepicker( "option", "appendText" );

// setter
$( ".selector" ).datepicker( "option", "appendText", "(yyyy-mm-dd)" );
```

autoSizeType: Boolean

Default: `false` 如果设置为 `true` ，将自动调整输入字段，以适应日期在当前的 `dateFormat` 。**Code examples:**

初始化带有指定 `autoSize` 选项的 datepicker：

```
$( ".selector" ).datepicker({ autoSize: true });
```

在初始化后，获取或设置 `autoSize` 选项：

```
// getter
var autoSize = $( ".selector" ).datepicker( "option", "autoSize" );

// setter
$( ".selector" ).datepicker( "option", "autoSize", true );
```

beforeShowType: Function(Element input, Object inst)

Default: `null` 一个函数，它接受一个输入字段和当前的datepicker实例， 并返回一个选项对象来修改datepicker。只在datepicker显示之前被调用。

beforeShowDayType: Function(Date date)

Default: `null` 一个函数，它接受一个日期作为参数 并且 必须返回一个数组：

- `[0]`：`true` / `false` 表示这个日期是否可选
- `[1]`：一个 CSS class 名 添加到日期的单元格 或默认描述为 `""`。
- `[2]`：一个可选的日期弹出提示

该函数在datepicker 每一天显示之前被调用。

buttonImageType: String

Default: `""` 当 `showOn` 选项设置为 `"button"` 或 `"both"` 时， 用于显示datepicker的图片url地址。 如果设置， `buttonText` 选项将成为 `alt` 值，而不是直接显示。**Code examples:**

初始化带有指定 `buttonImage` 选项的 datepicker：

```
$( ".selector" ).datepicker({ buttonImage: "/images/datepicker.gif" });
```

在初始化后，获取或设置 `buttonImage` 选项：

```
// getter
var buttonImage = $( ".selector" ).datepicker( "option", "buttonImage" );

// setter
$( ".selector" ).datepicker( "option", "buttonImage", "/images/datepicker.gif" );
```

buttonImageOnlyType: Boolean

Default: `false` 无论是按钮 图像 必须自行提供 而不是 里面一个按钮元素。 此选项仅当 `buttonImage` 选项也被设置时才起作用。**Code examples:**

初始化带有指定 `buttonImageOnly` 选项的 datepicker：

```
$( ".selector" ).datepicker({ buttonImageOnly: true });
```

在初始化后，获取或设置 `buttonImageOnly` 选项：

```
// getter
var buttonImageOnly = $( ".selector" ).datepicker( "option", "buttonImageOnly" );

// setter
$( ".selector" ).datepicker( "option", "buttonImageOnly", true );
```

buttonTextType: String

Default: `"..."` 显示在触发按钮上的文本。和 `showOn` 设置为 `"button"` 或 `"both"` 时结合使用。**Code examples:**

初始化带有指定 `buttonText` 选项的 datepicker：

```
$( ".selector" ).datepicker({ buttonText: "Choose" });
```

在初始化后，获取或设置 `buttonText` 选项：

```
// getter
var buttonText = $( ".selector" ).datepicker( "option", "buttonText" );

// setter
$( ".selector" ).datepicker( "option", "buttonText", "Choose" );
```

calculateWeekType: Function()

Default: `jQuery.datepicker.iso8601Week` 一个函数来计算当前周是一年中的第几周。 默认实现使用ISO8601的定义：的定义执行: 每周从周一开始；每年的第一个星期包含这年的第一个星期四。愚人码头注：这意味着今年的第一周中可能会包含去年的3天,并且今年的3天可能会被包含进去年的最后一周中。**Code examples:**

初始化带有指定 `calculateWeek` 选项的 datepicker：

```
$( ".selector" ).datepicker({ calculateWeek: myWeekCalc });
```

在初始化后，获取或设置 `calculateWeek` 选项：

```
// getter
var calculateWeek = $( ".selector" ).datepicker( "option", "calculateWeek" );

// setter
$( ".selector" ).datepicker( "option", "calculateWeek", myWeekCalc );
```

changeMonthType: Boolean

Default: `false` 允许你将月份修改为一个下拉菜单。 你可以将参数设置为false来禁用此功能，也就是显示为文字。**Code examples:**

初始化带有指定 `changeMonth` 选项的 datepicker：

```
$( ".selector" ).datepicker({ changeMonth: true });
```

在初始化后，获取或设置 `changeMonth` 选项：

```
// getter
var changeMonth = $( ".selector" ).datepicker( "option", "changeMonth" );

// setter
$( ".selector" ).datepicker( "option", "changeMonth", true );
```

changeYearType: Boolean

Default: `false` 允许你将年份修改为一个下拉菜单。 你可以将参数设置为false来禁用此功能，也就是显示为文字。使用 `yearRange` 选项来控制哪些年是可供选择。**Code examples:**

初始化带有指定 `changeYear` 选项的 datepicker：

```
$( ".selector" ).datepicker({ changeYear: true });
```

在初始化后，获取或设置 `changeYear` 选项：

```
// getter
var changeYear = $( ".selector" ).datepicker( "option", "changeYear" );

// setter
$( ".selector" ).datepicker( "option", "changeYear", true );
```

closeTextType: String

Default: `"Done"` 关闭连接显示的文字。使用 `showButtonPanel` 选项以显示此按钮。**Code examples:**

初始化带有指定 `closeText` 选项的 datepicker：

```
$( ".selector" ).datepicker({ closeText: "Close" });
```

在初始化后，获取或设置 `closeText` 选项：

```
// getter
var closeText = $( ".selector" ).datepicker( "option", "closeText" );

// setter
$( ".selector" ).datepicker( "option", "closeText", "Close" );
```

constrainInputType: Boolean

Default: `true` 当值为 `true` 时，在输入栏的输入被限制为当前的日期格式 `dateFormat` 选项允许的字符。**Code examples:**

初始化带有指定 `constrainInput` 选项的 datepicker：

```
$( ".selector" ).datepicker({ constrainInput: false });
```

在初始化后，获取或设置 `constrainInput` 选项：

```
// getter
var constrainInput = $( ".selector" ).datepicker( "option", "constrainInput" );

// setter
$( ".selector" ).datepicker( "option", "constrainInput", false );
```

currentTextType: String

Default: `"Today"` 当前日期链接以文本形式显示。使用 `showButtonPanel` 选项以显示此按钮。**Code examples:**

初始化带有指定 `currentText` 选项的 datepicker：

```
$( ".selector" ).datepicker({ currentText: "Now" });
```

在初始化后，获取或设置 `currentText` 选项：

```
// getter
var currentText = $( ".selector" ).datepicker( "option", "currentText" );

// setter
$( ".selector" ).datepicker( "option", "currentText", "Now" );
```

dateFormatType: String

Default: `"mm/dd/yy"` 描述和显示日期的格式。 对于可能的格式的完整列表，请参阅 `[` formatDate `](#utility-formatDate)` 函数。**Code examples:**

初始化带有指定 `dateFormat` 选项的 datepicker：

```
$( ".selector" ).datepicker({ dateFormat: "yy-mm-dd" });
```

在初始化后，获取或设置 `dateFormat` 选项：

```
// getter
var dateFormat = $( ".selector" ).datepicker( "option", "dateFormat" );

// setter
$( ".selector" ).datepicker( "option", "dateFormat", "yy-mm-dd" );
```

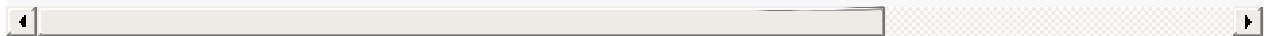
dayNamesType: Array

Default:

`[ "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday" ]` 日期长的名字的列表，从星期日(Sunday)开始，依照 `dateFormat` 的设置进行使用。**Code examples:**

初始化带有指定 `dayNames` 选项的 datepicker：

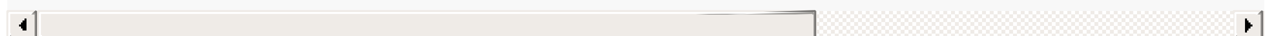
```
$( ".selector" ).datepicker({ dayNames: [ "Dimanche", "Lundi", "Mardi", "Mercredi", "Jeud
```



在初始化后，获取或设置 `dayNames` 选项：

```
// getter
var dayNames = $( ".selector" ).datepicker( "option", "dayNames" );

// setter
$( ".selector" ).datepicker( "option", "dayNames", [ "Dimanche", "Lundi", "Mardi", "Mercr
```

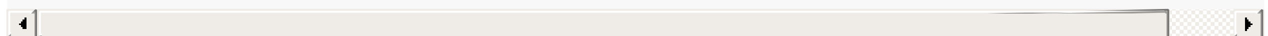


dayNamesMinType: Array

Default: `[ "Su", "Mo", "Tu", "We", "Th", "Fr", "Sa" ]` 日期最小化简称的列表, 从周日开始(Sunday), 用在datepicker每列的头部.。**Code examples:**

初始化带有指定 `dayNamesMin` 选项的 datepicker：

```
$( ".selector" ).datepicker({ dayNamesMin: [ "Di", "Lu", "Ma", "Me", "Je", "Ve", "Sa" ] }
```



在初始化后，获取或设置 `dayNamesMin` 选项：

```
// getter
var dayNamesMin = $( ".selector" ).datepicker( "option", "dayNamesMin" );

// setter
$( ".selector" ).datepicker( "option", "dayNamesMin", [ "Di", "Lu", "Ma", "Me", "Je", "Ve
```

dayNamesShortType: Array

Default: `[ "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" ]` 日期名称的简写的列表, 从周日开始(Sunday), 依照 `dateFormat` 的设置使用。**Code examples:**

初始化带有指定 `dayNamesShort` 选项的 datepicker：

```
$( ".selector" ).datepicker({ dayNamesShort: [ "Dim", "Lun", "Mar", "Mer", "Jeu", "Ven",
```

在初始化后，获取或设置 `dayNamesShort` 选项：

```
// getter
var dayNamesShort = $( ".selector" ).datepicker( "option", "dayNamesShort" );

// setter
$( ".selector" ).datepicker( "option", "dayNamesShort", [ "Dim", "Lun", "Mar", "Mer", "Je
```

defaultDateType: Date or Number or String

Default: `null` 如果该字段为空时设置第一次打开时显示的日期。 通过Date对象或字符串在当前 `dateFormat` 中指定任何一个实际的日期， 或者一个和今天对比的数字（例如 +7） 或者一个连续的字符串值('y' 表示年, 'm' 表示月, 'w'表示周, 'd'表示日, 例如. '+1m +7d'), 或者为空则是今天.支持多个类型：

- **Date**: 一个包含默认日期的date对象。
- **Number**: 一个和今天对比的数字。例如 `2` 表示从今天开始的第二天，（愚人码头注：即：后天）， `-1` 表示昨天。
- **String**: 一个由 `dateFormat` 选项定义格式的字符串， 或相对日期。 相对日期必须包含值和期间对; 有效期间为：`"y"` 表示几年， `"m"` 表示几月， `"w"` 表示几周，和 `"d"` 表示几天。 例如， `"+1m +7d"` 表示从今天开始的一个月加七天。

Code examples:

初始化带有指定 `defaultDate` 选项的 datepicker：

```
$( ".selector" ).datepicker({ defaultDate: +7 });
```

在初始化后，获取或设置 `defaultDate` 选项：

```
// getter
var defaultDate = $( ".selector" ).datepicker( "option", "defaultDate" );

// setter
$( ".selector" ).datepicker( "option", "defaultDate", +7 );
```

durationType: or String

Default: `"normal"` 设置datepicker展开动画的显示时间，可以是一个毫秒值, 也可以使用以下的三种字符值来表示("slow", "normal", "fast"), 或者为 "则马上显示。**Code examples:**

初始化带有指定 `duration` 选项的 datepicker：

```
$( ".selector" ).datepicker({ duration: "slow" });
```

在初始化后，获取或设置 `duration` 选项：

```
// getter
var duration = $( ".selector" ).datepicker( "option", "duration" );

// setter
$( ".selector" ).datepicker( "option", "duration", "slow" );
```

firstDayType: Integer

Default: `0` 设置一周中的第一天:周日是 `0` , 周一是 `1` , 以此类推。**Code examples:**

初始化带有指定 `firstDay` 选项的 datepicker：

```
$( ".selector" ).datepicker({ firstDay: 1 });
```

在初始化后，获取或设置 `firstDay` 选项：

```
// getter
var firstDay = $( ".selector" ).datepicker( "option", "firstDay" );

// setter
$( ".selector" ).datepicker( "option", "firstDay", 1 );
```

gotoCurrentType: Boolean

Default: `false` 如果设置为 `true` , 当前日的链接将移动到当前选中的日期，而不是今天。**Code examples:**

初始化带有指定 `gotoCurrent` 选项的 datepicker：

```
$( ".selector" ).datepicker({ gotoCurrent: true });
```

在初始化后，获取或设置 `gotoCurrent` 选项：

```
// getter
var gotoCurrent = $( ".selector" ).datepicker( "option", "gotoCurrent" );

// setter
$( ".selector" ).datepicker( "option", "gotoCurrent", true );
```

hideIfNoPrevNextType: Boolean

Default: `false` 通常当前一个和下一个链接被禁用时不适用 (参看 `minDate` 和 `maxDate`). 你可以通过设置此属性为 `true` 完全的隐藏他们.**Code examples:**

初始化带有指定 `hideIfNoPrevNext` 选项的 datepicker：

```
$( ".selector" ).datepicker({ hideIfNoPrevNext: true });
```

在初始化后，获取或设置 `hideIfNoPrevNext` 选项：

```
// getter
var hideIfNoPrevNext = $( ".selector" ).datepicker( "option", "hideIfNoPrevNext" );

// setter
$( ".selector" ).datepicker( "option", "hideIfNoPrevNext", true );
```

isRTLType: Boolean

Default: `false` 当前语言描述是否为自右向左的。（愚人码头注：例如阿拉伯语）**Code examples:**

初始化带有指定 `isRTL` 选项的 datepicker：

```
$( ".selector" ).datepicker({ isRTL: true });
```

在初始化后，获取或设置 `isRTL` 选项：

```
// getter
var isRTL = $( ".selector" ).datepicker( "option", "isRTL" );

// setter
$( ".selector" ).datepicker( "option", "isRTL", true );
```

maxDateType: Date or Number or String

Default: `null` 最大的可选日期。当设置为 `null` 时，没有上限。支持多个类型：

- **Date**: 一个包含默认日期的date对象。
- **Number**: 一个和今天对比的数字。例如 `2` 表示从今天开始的第二天，（愚人码头注：即：后天）， `-1` 表示昨天。
- **String**: 一个由 `dateFormat` 选项定义格式的字符串， 或相对日期。 相对日期必须包含值和期间对; 有效期间为： `"y"` 表示几年， `"m"` 表示几月， `"w"` 表示几周，和 `"d"` 表示几天。 例如， `"+1m +7d"` 表示从今天开始的一个月加七天。

Code examples:

初始化带有指定 `maxDate` 选项的 datepicker：

```
$( ".selector" ).datepicker({ maxDate: "+1m +1w" });
```

在初始化后，获取或设置 `maxDate` 选项：

```
// getter
var maxDate = $( ".selector" ).datepicker( "option", "maxDate" );

// setter
$( ".selector" ).datepicker( "option", "maxDate", "+1m +1w" );
```

minDateType: Date or Number or String

Default: `null` 最小的可选日期。当设置为 `null` 时，没有下限。支持多个类型：

- **Date**: 一个包含默认日期的date对象。
- **Number**: 一个和今天对比的数字。例如 `2` 表示从今天开始的第二天，（愚人码头注：即：后天）， `-1` 表示昨天。
- **String**: 一个由 `dateFormat` 选项定义格式的字符串， 或相对日期。 相对日期必须包含值和期间对; 有效期间为： `"y"` 表示几年， `"m"` 表示几月， `"w"` 表示几周，和 `"d"` 表示几天。 例如， `"+1m +7d"` 表示从今天开始的一个月加七天。

Code examples:

初始化带有指定 `minDate` 选项的 datepicker：

```
$( ".selector" ).datepicker({ minDate: new Date(2007, 1 - 1, 1) });
```

在初始化后，获取或设置 `minDate` 选项：

```
// getter
var minDate = $( ".selector" ).datepicker( "option", "minDate" );

// setter
$( ".selector" ).datepicker( "option", "minDate", new Date(2007, 1 - 1, 1) );
```

monthNamesType: Array

Default:

`[ "January", "February", "March", "April", "May", "June", "July", "August", "September", "(`

月份的完全名称列表，依照 `dateFormat` 的设置进行使用。**Code examples:**

初始化带有指定 `monthNames` 选项的 datepicker：

```
$( ".selector" ).datepicker({ monthNames: [ "Januar", "Februar", "Marts", "April", "Maj",
```



在初始化后，获取或设置 `monthNames` 选项：

```
// getter
var monthNames = $( ".selector" ).datepicker( "option", "monthNames" );

// setter
$( ".selector" ).datepicker( "option", "monthNames", [ "Januar", "Februar", "Marts", "Apr
```



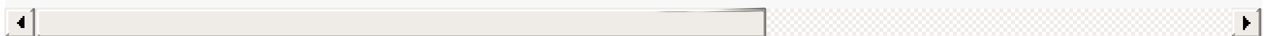
monthNamesShortType: Array

Default:

`[ "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" ]` 月份简写名称的列表，依照 `dateFormat` 的设置进行使用。**Code examples:**

初始化带有指定 `monthNamesShort` 选项的 datepicker：

```
$( ".selector" ).datepicker({ monthNamesShort: [ "Jan", "Feb", "Mar", "Apr", "Maj", "Jun"
```



在初始化后，获取或设置 `monthNamesShort` 选项：

```
// getter
var monthNamesShort = $( ".selector" ).datepicker( "option", "monthNamesShort" );

// setter
$( ".selector" ).datepicker( "option", "monthNamesShort", [ "Jan", "Feb", "Mar", "Apr", "
```



navigationAsDateFormatType: Boolean

Default: `false` 确定是 `prevText` 和 `nextText` 选项是否应该被解析为 `[` formatDate `](#utility-formatDate)` 函数的日期，让他们能够显示目标的月份名称.**Code examples:**

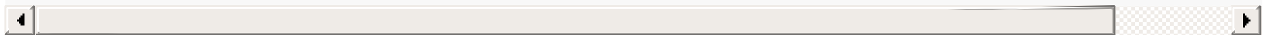
初始化带有指定 `navigationAsDateFormat` 选项的 datepicker：

```
$( ".selector" ).datepicker({ navigationAsDateFormat: true });
```

在初始化后，获取或设置 `navigationAsDateFormat` 选项：

```
// getter
var navigationAsDateFormat = $( ".selector" ).datepicker( "option", "navigationAsDateForm

// setter
$( ".selector" ).datepicker( "option", "navigationAsDateFormat", true );
```



nextTextType: String

Default: `"Next"` 下个月链接显示的文字。使用标准ThemeRoller样式， 这个值被替换为一个图标。**Code examples:**

初始化带有指定 `nextText` 选项的 datepicker：

```
$( ".selector" ).datepicker({ nextText: "Later" });
```

在初始化后，获取或设置 `nextText` 选项：

```
// getter
var nextText = $( ".selector" ).datepicker( "option", "nextText" );

// setter
$( ".selector" ).datepicker( "option", "nextText", "Later" );
```

numberOfMonthsType: Number or Array

Default: `1` 设置一次显示几个月.支持多个类型：

- **Number**: 一行显示的月份数
- **Array**: 一个数组定义的显示行数和列数。

Code examples:

初始化带有指定 `numberOfMonths` 选项的 datepicker：

```
$( ".selector" ).datepicker({ numberOfMonths: [ 2, 3 ] });
```

在初始化后，获取或设置 `numberOfMonths` 选项：

```
// getter
var numberOfMonths = $( ".selector" ).datepicker( "option", "numberOfMonths" );

// setter
$( ".selector" ).datepicker( "option", "numberOfMonths", [ 2, 3 ] );
```

onChangeMonthYearType: Function(Integer year, Integer month, Object inst)

Default: `null` 当datepicker移动到一个新的月份 并/或 年份时调用。 该函数接收选定的年份， 月份（1-12）， 和datepicker实例作为参数。 `this` 指向相关联的输入域。

onCloseType: Function(String dateText, Object inst)

Default: `null` 当datepicker关闭时调用，确定一个日期是否被选中。 该函数接收所选日期的文本（ `""` 如果没有）和datepicker实例作为参数。 `this` 指向相关联的输入域。

onSelectType: Function(String dateText, Object inst)

Default: `null` 当选择datepicker调用。 该函数接收所选日期的文本和datepicker实例作为参数。 `this` 指向相关联的输入域。

prevTextType: String

Default: `"Prev"` 上个月链接显示的文字 。使用标准ThemeRoller样式， 这个值被替换为一个图标。**Code examples:**

初始化带有指定 `prevText` 选项的 datepicker：

```
$( ".selector" ).datepicker({ prevText: "Earlier" });
```

在初始化后，获取或设置 `prevText` 选项：

```
// getter
var prevText = $( ".selector" ).datepicker( "option", "prevText" );

// setter
$( ".selector" ).datepicker( "option", "prevText", "Earlier" );
```

selectOtherMonthsType: Boolean

Default: `false` 显示在当前月份的之前或之后的日期是否可以被选择。（愚人码头注：比如2014年4月1日是周二，如果 `showOtherMonths` 选项设置为 `true` ，那么2014年3月31日，2014年3月30日会以灰色的形式显示在4月份的面板上，这个选项表示在4月份的面板上是否可以选择2014年3月31日，2014年3月30日这两个日期。5月也是一样的。） 这选项仅适用于 如果 `showOtherMonths` 选项设置为 `true` 的时候。**Code examples:**

初始化带有指定 `selectOtherMonths` 选项的 datepicker：

```
$( ".selector" ).datepicker({ selectOtherMonths: true });
```

在初始化后，获取或设置 `selectOtherMonths` 选项：

```
// getter
var selectOtherMonths = $( ".selector" ).datepicker( "option", "selectOtherMonths" );

// setter
$( ".selector" ).datepicker( "option", "selectOtherMonths", true );
```

shortYearCutoffType: Number or String

Default: `"+10"` 设置一个定义日期所处实际的截断年份值(与 `dateFormat` 中的 'y'共同使用). 如果是一个数值 (0-99)那么将直接使用这些值. 如果提供的是一个字符串值那么它将被转换为数值添加到当前年. 一旦截断年开始计算, 任何输入的日期的年份小于或者等于它的话将被判定为在本世纪,大于他的将被判定为在上个世纪.支持多个类型：

- **Number**: 一个 `0` 到 `99` 的值表示截断年份值
- **String**: 从本年份开始的相对年数，例如，`"+3"` or `"-5"` .

Code examples:

初始化带有指定 `shortYearCutoff` 选项的 datepicker：

```
$( ".selector" ).datepicker({ shortYearCutoff: 50 });
```

在初始化后，获取或设置 `shortYearCutoff` 选项：

```
// getter
var shortYearCutoff = $( ".selector" ).datepicker( "option", "shortYearCutoff" );

// setter
$( ".selector" ).datepicker( "option", "shortYearCutoff", 50 );
```

showAnimType: String

Default: `"show"` 设置显示/隐藏datepicker的动画名称. 使用 `"show"` (默认), `"slideDown"`, `"fadeIn"`, 或其他任何jQuery UI effects的显示/隐藏效果。 设置为空字符串可以禁用动画, 即直接显示或者隐藏。**Code examples:**

初始化带有指定 `showAnim` 选项的 datepicker：

```
$( ".selector" ).datepicker({ showAnim: "fold" });
```

在初始化后，获取或设置 `showAnim` 选项：

```
// getter
var showAnim = $( ".selector" ).datepicker( "option", "showAnim" );

// setter
$( ".selector" ).datepicker( "option", "showAnim", "fold" );
```

showButtonPanelType: Boolean

Default: `false` 是否显示日历下方的按钮面板。 按钮面板包含两个按钮， 一个今天按钮链接到当前日期， 和一个完成按钮关闭datepicker。 该按钮的文本可以使用 `currentText` 和 `closeText` 选项分别进行定制。**Code examples:**

初始化带有指定 `showButtonPanel` 选项的 datepicker：

```
$( ".selector" ).datepicker({ showButtonPanel: true });
```

在初始化后，获取或设置 `showButtonPanel` 选项：

```
// getter
var showButtonPanel = $( ".selector" ).datepicker( "option", "showButtonPanel" );

// setter
$( ".selector" ).datepicker( "option", "showButtonPanel", true );
```

showCurrentAtPosType: Number

Default: `0` 通过 `numberOfMonths` 选项设置多月份显示的情况下，当前月份显示的位置。（愚人码头注：自顶部/左边开始第n位，以0开始计数。）**Code examples:**

初始化带有指定 `showCurrentAtPos` 选项的 datepicker：

```
$( ".selector" ).datepicker({ showCurrentAtPos: 3 });
```

在初始化后，获取或设置 `showCurrentAtPos` 选项：

```
// getter
var showCurrentAtPos = $( ".selector" ).datepicker( "option", "showCurrentAtPos" );

// setter
$( ".selector" ).datepicker( "option", "showCurrentAtPos", 3 );
```

showMonthAfterYearType: Boolean

Default: `false` 是否在面板的头部年份后面显示月份。**Code examples:**

初始化带有指定 `showMonthAfterYear` 选项的 datepicker：

```
$( ".selector" ).datepicker({ showMonthAfterYear: true });
```

在初始化后，获取或设置 `showMonthAfterYear` 选项：

```
// getter
var showMonthAfterYear = $( ".selector" ).datepicker( "option", "showMonthAfterYear" );

// setter
$( ".selector" ).datepicker( "option", "showMonthAfterYear", true );
```

showOnType: String

Default: `"focus"` 设置触发datepicker自动出现的事件名称,是focus (`"focus"`)还是clicked (`"button"`)或者任何事件(`"both"`)。**Code examples:**

初始化带有指定 `showOn` 选项的 datepicker：

```
$( ".selector" ).datepicker({ showOn: "both" });
```

在初始化后，获取或设置 `showOn` 选项：

```
// getter
var showOn = $( ".selector" ).datepicker( "option", "showOn" );

// setter
$( ".selector" ).datepicker( "option", "showOn", "both" );
```

showOptionsType: Object

Default: `{}` 如果想 `showAnim` 选项来使用jQuery UI effects动画效果, 你可以为动画提供一些额外的设置.**Code examples:**

初始化带有指定 `showOptions` 选项的 datepicker：

```
$( ".selector" ).datepicker({ showOptions: { direction: "up" } });
```

在初始化后，获取或设置 `showOptions` 选项：

```
// getter
var showOptions = $( ".selector" ).datepicker( "option", "showOptions" );

// setter
$( ".selector" ).datepicker( "option", "showOptions", { direction: "up" } );
```

showOtherMonthsType: Boolean

Default: `false` 是否在当前月份面板显示上、下两个月的一些日期数（不可选）。 想要让这些日期可选，请使用 `selectOtherMonths` 选项。**Code examples:**

初始化带有指定 `showOtherMonths` 选项的 datepicker：

```
$( ".selector" ).datepicker({ showOtherMonths: true });
```

在初始化后，获取或设置 `showOtherMonths` 选项：

```
// getter
var showOtherMonths = $( ".selector" ).datepicker( "option", "showOtherMonths" );

// setter
$( ".selector" ).datepicker( "option", "showOtherMonths", true );
```

showWeekType: Boolean

Default: `false` 如果为 `true` ，面板将增加一列，显示一年中的哪一周。 该 `calculateWeek` 选项决定一年中的哪一周是如何计算的。 您可以改变 `firstDay` 选项。**Code examples:**

初始化带有指定 `showWeek` 选项的 datepicker：

```
$( ".selector" ).datepicker({ showWeek: true });
```

在初始化后，获取或设置 `showWeek` 选项：

```
// getter
var showWeek = $( ".selector" ).datepicker( "option", "showWeek" );

// setter
$( ".selector" ).datepicker( "option", "showWeek", true );
```

stepMonthsType: Number

Default: `1` 当点击上/下一月链接时，一次翻几个月。**Code examples:**

初始化带有指定 `stepMonths` 选项的 datepicker：

```
$( ".selector" ).datepicker({ stepMonths: 3 });
```

在初始化后，获取或设置 `stepMonths` 选项：

```
// getter
var stepMonths = $( ".selector" ).datepicker( "option", "stepMonths" );

// setter
$( ".selector" ).datepicker( "option", "stepMonths", 3 );
```

weekHeaderType: String

Default: `"Wk"` 本年度哪一周 列的标题显示文本。 使用 `showWeek` 选项以显示此列。**Code examples:**

初始化带有指定 `weekHeader` 选项的 datepicker：

```
$( ".selector" ).datepicker({ weekHeader: "W" });
```

在初始化后，获取或设置 `weekHeader` 选项：

```
// getter
var weekHeader = $( ".selector" ).datepicker( "option", "weekHeader" );

// setter
$( ".selector" ).datepicker( "option", "weekHeader", "W" );
```

yearRangeType: String

Default: `"c-10:c+10"` 控制年份的下拉列表中显示的年份数量，可以是相对当前年(-nn:+nn)，相对于选择年份(-nn:+nn)，也可以是绝对值(`"nnnn:nnnn"`)，或这些格式的组合(`"nnnn:-nn"`)。 请注意，此选项仅影响下拉列表中的显示， 使用 `minDate` 和/或 `maxDate` 选项限制哪些日期可以选择。**Code examples:**

初始化带有指定 `yearRange` 选项的 datepicker：

```
$( ".selector" ).datepicker({ yearRange: "2002:2012" });
```

在初始化后，获取或设置 `yearRange` 选项：

```
// getter
var yearRange = $( ".selector" ).datepicker( "option", "yearRange" );

// setter
$( ".selector" ).datepicker( "option", "yearRange", "2002:2012" );
```

yearSuffixType: String

Default: `""` 显示在月份头部中年份后面的文本。**Code examples:**

初始化带有指定 `yearSuffix` 选项的 datepicker：

```
$( ".selector" ).datepicker({ yearSuffix: "CE" });
```

在初始化后，获取或设置 `yearSuffix` 选项：

```
// getter
var yearSuffix = $( ".selector" ).datepicker( "option", "yearSuffix" );

// setter
$( ".selector" ).datepicker( "option", "yearSuffix", "CE" );
```

Methods

destroy()Returns: jQuery (plugin only)

完全移除datepicker功能. 这将使元素返回到之前的初始化状态.

- 该方法不接受任何参数。

Code examples:

调用 destroy 方法：

```
$( ".selector" ).datepicker( "destroy" );
```

dialog(date [, onSelect] [, settings] [, pos])Returns: jQuery (plugin only)

在一个"dialog"中打开一个datepicker。

- **date**Type: String or Date初始化的日期。
- **onSelect**Type: Function()当一个日期被选中时的回调函数. 这个函数接收日期文本 和datepicker实例作为参数。
- **settings**Type: Options新的datepicker的选项。

- **pos**Type: Number[2] or MouseEventdialog对话框的top/left的 `[x, y]` 坐标，或者一个鼠标事件 `MouseEvent` 的坐标。如果不提供此参数dialog将显示在屏幕正中。

Code examples:

调用 dialog 方法：

```
$( ".selector" ).datepicker( "dialog", "10/12/2012" );
```

getDate()Returns: Date

返回一个datepicker中当前日期, 如果没有日期被选中的话那么返回 `null` 。

- 该方法不接受任何参数。

Code examples:

调用 getDate 方法：

```
var currentDate = $( ".selector" ).datepicker( "getDate" );
```

hide()Returns: jQuery (plugin only)

关闭先前打开的date picker。

- 该方法不接受任何参数。

Code examples:

调用 hide 方法：

```
$( ".selector" ).datepicker( "hide" );
```

isDisabled()Returns: Boolean

确定一个datepicker是否已禁用.

- 该方法不接受任何参数。

Code examples:

调用 isDisabled 方法：

```
var isDisabled = $( ".selector" ).datepicker( "isDisabled" );
```

option(optionName)Returns: Object

获取当前与指定的 `optionName` 关联的值。

- **optionName**Type: String要获取值的选项的名称。

Code examples:

调用该方法：

```
var isDisabled = $( ".selector" ).datepicker( "option", "disabled" );
```

option()Returns: PlainObject

获取一个包含键/值对的对象，键/值对表示当前 datepicker 选项哈希。

- 该方法不接受任何参数。

Code examples:

调用该方法：

```
var options = $( ".selector" ).datepicker( "option" );
```

option(optionName, value)Returns: jQuery (plugin only)

设置与指定的 `optionName` 关联的 datepicker 选项的值。

- **optionName**Type: String要设置的选项的名称。
- **value**Type: Object要为选项设置的值。

Code examples:

调用该方法：

```
$( ".selector" ).datepicker( "option", "disabled", true );
```

option(options)Returns: jQuery (plugin only)

为 datepicker 设置一个或多个选项。

- **options**Type: Object要设置的 option-value 对。

Code examples:

调用该方法：

```
$( ".selector" ).datepicker( "option", { disabled: true } );
```

refresh()Returns: jQuery (plugin only)

在作出一些外部修改后，重绘日期选择器。

- 该方法不接受任何参数。

Code examples:

调用 refresh 方法：

```
$( ".selector" ).datepicker( "refresh" );
```

setDate(date)Returns: jQuery (plugin only)

为datepicker设置日期。 新的日期可能是一个 `Date` 对象或当前date format 的字符串（例如， `"01/26/2009"` ）， 一个从今天开始的天数（例如， `+7` ） 或值和周期的字符串 `"y"` 表示年数, `"m"` 表示月份数, `"w"` 表示周数, `"d"` 表示天数,例如， `"+1m +7d"`)， 或者为 `null` 来清除选定的日期。

- **date**Type: String or Date新的日期。

Code examples:

调用 setDate 方法：

```
$( ".selector" ).datepicker( "setDate", "10/12/2012" );
```

show()Returns: jQuery (plugin only)

显示日期datepicker插件。 如果datepicker被绑定到input元素， 要显示datepicker的input元素必须是可见的。

- 该方法不接受任何参数。

Code examples:

调用 show 方法：

```
$( ".selector" ).datepicker( "show" );
```

widget()Returns: jQuery

返回一个包含 datepicker 的 `jQuery` 对象。

- 该方法不接受任何参数。

Code examples:

调用 widget 方法：

```
var widget = $( ".selector" ).datepicker( "widget" );
```

Example:

一个简单的 jQuery UI Datepicker.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>datepicker demo</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-ui.cs
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="datepicker"></div>

<script>
$( "#datepicker" ).datepicker();
</script>

</body>
</html>
```

Dialog Widget

Categories: Widgets

version added: 1.0

Description: 在一个交互覆盖层中打开内容。

QuickNavExamples

Options

- appendTo
- autoOpen
- buttons
- closeOnEscape
- closeText
- dialogClass
- draggable
- height
- hide
- maxHeight
- maxWidth
- minHeight
- minWidth
- modal
- position
- resizable
- show
- title
- width

Methods

- close
- destroy
- isOpen
- moveToTop

- open
- option
- widget

Extension Points

- _allowInteraction

Events

- beforeClose
- close
- create
- drag
- dragStart
- dragStop
- focus
- open
- resize
- resizeStart
- resizeStop

对话框是一个悬浮窗口，包括一个标题栏和一个内容区域。对话框窗口可以移动，重新调整大小，默认情况下通过 'x' 图标关闭。

如果内容长度超过最大高度，一个滚动条会自动出现。

一个底部按钮栏和一个半透明的模式覆盖层是常见的添加选项。

焦点

当打开一个对话框时，焦点会自动移动到满足下面条件的第一个项目：

1. 带有 `autofocus` 属性的对话框内的第一个元素
2. 对话框内容内的第一个 `:tabbable` 元素
3. 对话框按钮面板内的第一个 `:tabbable` 元素
4. 对话框的关闭按钮
5. 对话框本身

当打开时，对话框部件（Dialog Widget）确保通过 tab 切换对话框内元素间的焦点，不包括对话框外的元素。模态对话框防止鼠标用户点击对话框外的元素。

当关闭对话框时，焦点自动返回到之前对话框打开时获得焦点的元素上。

隐藏关闭按钮

在一些情况下，您可能想要隐藏关闭按钮，例如，在按钮面板中已经有一个关闭按钮的情况。最好的解决方法是通过 CSS。作为实例，您可以定义一个简单的规则，比如：

```
.no-close .ui-dialog-titlebar-close {
  display: none;
}
```

然后，您可以添加 `no-close` class 到任意的对话框，用来隐藏关闭按钮：

```
$( "#dialog" ).dialog({
  dialogClass: "no-close",
  buttons: [
    {
      text: "OK",
      click: function() {
        $( this ).dialog( "close" );
      }
    }
  ]
});
```

主题

对话框部件（Dialog Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用对话框指定的样式，则可以使用下面的 CSS class 名称：

- `ui-dialog`：对话框的外层容器。
  - `ui-dialog-titlebar`：包含对话框标题和关闭按钮的标题栏。
    - `ui-dialog-title`：对话框文本标题周围的容器。
    - `ui-dialog-titlebar-close`：对话框的关闭按钮。
  - `ui-dialog-content`：对话框内容周围的容器。这也是部件被实例化的元素。
  - `ui-dialog-buttonpane`：包含对话按钮的面板。只有当设置了 `buttons` 选项时才呈现。
    - `ui-dialog-buttonset`：按钮周围的容器。

此外，当设置了 `modal` 选项时，一个带有 `ui-widget-overlay` class 名称的元素被追加到 `<body>`。

依赖

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 定位（Position）
- 按钮部件（Button Widget）
- 可拖拽小部件（Draggable Widget）（可选的；当与 `draggable` 选项一起使用时）

- 可调整尺寸小部件（Resizable Widget）（可选的；当与 `resizable` 选项一起使用时）
- 特效核心（Effects Core）（可选的；当与 `show` 和 `hide` 选项一起使用时）

其他注意事项：

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

Options

appendToType: Selector

Default: `"body"`

dialog（和遮罩层，如果modal存在）应该被追加到哪个元素。

注意:当dialog处于打开状态的时候 `appendTo` 选项不应该被改变。(version added: 1.10.0)**Code examples:**

初始化带有指定 `appendTo` 选项的 dialog：

```
$( ".selector" ).dialog({ appendTo: "#someElem" });
```

在初始化后，获取或设置 `appendTo` 选项：

```
// getter
var appendTo = $( ".selector" ).dialog( "option", "appendTo" );

// setter
$( ".selector" ).dialog( "option", "appendTo", "#someElem" );
```

autoOpenType: Boolean

Default: `true` 当设置为 `true` 时， dialog 会在初始化时自动打开. 如果为 `false` dialog 将会继续隐藏直到调用 `open()` 方法 。**Code examples:**

初始化带有指定 `autoOpen` 选项的 dialog：

```
$( ".selector" ).dialog({ autoOpen: false });
```

在初始化后，获取或设置 `autoOpen` 选项：

```
// getter
var autoOpen = $( ".selector" ).dialog( "option", "autoOpen" );

// setter
$( ".selector" ).dialog( "option", "autoOpen", false );
```

buttonsType: Object or Array

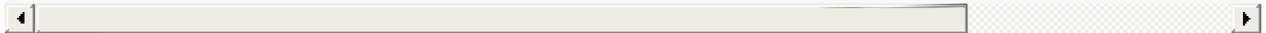
Default: `{}` 指定哪些按钮应该在dialog上显示。 回调的上下文是dialog元素（愚人码头注： `this` 指向）；如果你需要访问按钮， 可以利用事件对象的目标元素。支持多个类型：

- **Object**: 键是按钮标签，值是点击相关按钮时执行的回调函数。
- **Array**: 该数组的每个元素必须是 一个定义特性，属性，和按钮上设置的事件处理程序的对象。

Code examples:

初始化带有指定 `buttons` 选项的 dialog：

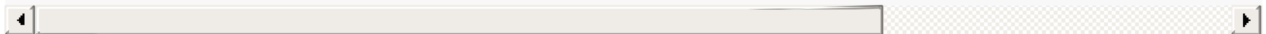
```
$( ".selector" ).dialog({ buttons: [ { text: "Ok", click: function() { $( this ).dialog(
```



在初始化后，获取或设置 `buttons` 选项：

```
// getter
var buttons = $( ".selector" ).dialog( "option", "buttons" );

// setter
$( ".selector" ).dialog( "option", "buttons", [ { text: "Ok", click: function() { $( this
```



closeOnEscapeType: Boolean

Default: `true` 指定具有焦点的dialog，在用户按下退出（ESC）键时，是否应该关闭。**Code examples:**

初始化带有指定 `closeOnEscape` 选项的 dialog：

```
$( ".selector" ).dialog({ closeOnEscape: false });
```

在初始化后，获取或设置 `closeOnEscape` 选项：

```
// getter
var closeOnEscape = $( ".selector" ).dialog( "option", "closeOnEscape" );

// setter
$( ".selector" ).dialog( "option", "closeOnEscape", false );
```

closeTextType: String

Default: `"close"` 指定关闭按钮的文本。 注意，关闭文本在使用标准的主题时，是隐藏的（visibly：hidden）。**Code examples:**

初始化带有指定 `closeText` 选项的 dialog：

```
$( ".selector" ).dialog({ closeText: "hide" });
```

在初始化后，获取或设置 `closeText` 选项：

```
// getter
var closeText = $( ".selector" ).dialog( "option", "closeText" );

// setter
$( ".selector" ).dialog( "option", "closeText", "hide" );
```

dialogClassType: String

Default: `""` 在使用额外附加的主题时，指定dialog的类名称，这些样式添加到dialog上。**Code examples:**

初始化带有指定 `dialogClass` 选项的 dialog：

```
$( ".selector" ).dialog({ dialogClass: "alert" });
```

在初始化后，获取或设置 `dialogClass` 选项：

```
// getter
var dialogClass = $( ".selector" ).dialog( "option", "dialogClass" );

// setter
$( ".selector" ).dialog( "option", "dialogClass", "alert" );
```

draggableType: Boolean

Default: `true` 如果设置为 `true`，dialog将可以使用标题栏实现拖动。需要包含 jQuery UI Draggable 部件。**Code examples:**

初始化带有指定 `draggable` 选项的 dialog：

```
$( ".selector" ).dialog({ draggable: false });
```

在初始化后，获取或设置 `draggable` 选项：

```
// getter
var draggable = $( ".selector" ).dialog( "option", "draggable" );

// setter
$( ".selector" ).dialog( "option", "draggable", false );
```

heightType: Number or String

Default: `"auto"` 设置对话框的高度（单位：像素）。支持多个类型：

- **Number**: 高度，单位：像素。
- **String**: 唯一支持的字符串值为 `"auto"`，这将使对话框高度根据其内容进行自动调整。

Code examples:

初始化带有指定 `height` 选项的 dialog：

```
$( ".selector" ).dialog({ height: 400 });
```

在初始化后，获取或设置 `height` 选项：

```
// getter
var height = $( ".selector" ).dialog( "option", "height" );

// setter
$( ".selector" ).dialog( "option", "height", 400 );
```

hideType: Boolean or Number or String or Object

Default: `null` dialog关闭（隐藏）时的动画效果。支持多个类型：

- **Boolean**: 当设置为 `false`， 将不使用动画效果，该dialog会立即被隐藏。 如果设置为 `true`， 该dialog将会以默认的持续时间和默认的效果淡出。
- **Number**: 该dialog将以指定的时间和默认的效果淡出。
- **String**: 该dialog将使用指定的效果被隐藏。 该值可以是一个jQuery内置的动画方法的名称， 如 `"slideUp"`， 或一个 jQuery UI 效果的名称， 如 `"fold"`。 在这两种情况下，将使用默认持续时间和默认的动画效果。
- **Object**: 如果该值是一个对象， 那么 `effect`, `delay`, `duration`, 和 `easing` 可能要提供。 如果 `effect` 属性包含一个jQuery方法的名称， 那么该方法将被使用; 否则它被假定为是一个jQuery UI的效果的名称。 当使用jQuery UI 支持额外设置 的效果， 你可以在对象中包含那些设置 并且它们将被传递到的效果。如果 `duration` 持续时间或 `easing` 属性被省略， 那么默认值将被使用。 如果 `effect` 被省略， 那么 `"fadeOut"` 将被使用。如果 `delay` 被省略， 那么将不使用延迟。

Code examples:

初始化带有指定 `hide` 选项的 dialog：

```
$( ".selector" ).dialog({ hide: { effect: "explode", duration: 1000 } });
```

在初始化后，获取或设置 `hide` 选项：

```
// getter
var hide = $( ".selector" ).dialog( "option", "hide" );

// setter
$( ".selector" ).dialog( "option", "hide", { effect: "explode", duration: 1000 } );
```

maxHeightType: Number

Default: `false` dialog可以调整的最大高度，以像素为单位。**Code examples:**

初始化带有指定 `maxHeight` 选项的 dialog：

```
$( ".selector" ).dialog({ maxHeight: 600 });
```

在初始化后，获取或设置 `maxHeight` 选项：

```
// getter
var maxHeight = $( ".selector" ).dialog( "option", "maxHeight" );

// setter
$( ".selector" ).dialog( "option", "maxHeight", 600 );
```

maxWidthType: Number

Default: `false` dialog可以调整的最大宽度，以像素为单位。**Code examples:**

初始化带有指定 `maxWidth` 选项的 dialog：

```
$( ".selector" ).dialog({ maxWidth: 600 });
```

在初始化后，获取或设置 `maxWidth` 选项：

```
// getter
var maxWidth = $( ".selector" ).dialog( "option", "maxWidth" );

// setter
$( ".selector" ).dialog( "option", "maxWidth", 600 );
```

minHeightType: Number

Default: `150` dialog可以调整的最小高度，以像素为单位。**Code examples:**

初始化带有指定 `minHeight` 选项的 dialog：

```
$( ".selector" ).dialog({ minHeight: 200 });
```

在初始化后，获取或设置 `minHeight` 选项：

```
// getter
var minHeight = $( ".selector" ).dialog( "option", "minHeight" );

// setter
$( ".selector" ).dialog( "option", "minHeight", 200 );
```

minWidthType: Number

Default: `150` dialog可以调整的最小宽度，以像素为单位。**Code examples:**

初始化带有指定 `minWidth` 选项的 dialog：

```
$( ".selector" ).dialog({ minWidth: 200 });
```

在初始化后，获取或设置 `minWidth` 选项：

```
// getter
var minWidth = $( ".selector" ).dialog( "option", "minWidth" );

// setter
$( ".selector" ).dialog( "option", "minWidth", 200 );
```

modalType: Boolean

Default: `false` 如果设置为 `true`，该dialog将会有遮罩层；页面上的其他项目将被禁用，即，不能交互。遮罩层创建对话框下方，但高于其它页面元素。**Code examples:**

初始化带有指定 `modal` 选项的 dialog：

```
$( ".selector" ).dialog({ modal: true });
```

在初始化后，获取或设置 `modal` 选项：

```
// getter
var modal = $( ".selector" ).dialog( "option", "modal" );

// setter
$( ".selector" ).dialog( "option", "modal", true );
```

positionType: Object or String or Array

Default: `{ my: "center", at: "center", of: window }`

指定dialog显示的位置。该dialog将会处理冲突，使得尽可能多的dialog尽可能地可见。

注意: 不赞成使用 `String` 和 `Array` 格式。

支持多个类型：

- **Object**: 确定dialog打开时的位置。 `of` 选项默认为窗口， 但您可以指定其他元素定位。你可以参考jQuery UI Position实用工具，了解各种选项的更多细节。
- **String**:一个字符串，表示可视区内的位置。可能的值： `"center"`, `"left"`, `"right"`, `"top"`, `"bottom"`.
- **Array**: 一个包含相对于可见区域左上角*x*, *y*偏移坐标（单位为像素） 的数组， 或 一个可能值的字符串名称。

Code examples:

初始化带有指定 `position` 选项的 dialog：

```
$( ".selector" ).dialog({ position: { my: "left top", at: "left bottom", of: button } });
```

在初始化后，获取或设置 `position` 选项：

```
// getter
var position = $( ".selector" ).dialog( "option", "position" );

// setter
$( ".selector" ).dialog( "option", "position", { my: "left top", at: "left bottom", of: b
```

resizableType: Boolean

Default: `true` 如果设置为 `true`， 那么dialog允许调整大小。需要包含jQuery UI Resizable widget。**Code examples:**

初始化带有指定 `resizable` 选项的 dialog：

```
$( ".selector" ).dialog({ resizable: false });
```

在初始化后，获取或设置 `resizable` 选项：

```
// getter
var resizable = $( ".selector" ).dialog( "option", "resizable" );

// setter
$( ".selector" ).dialog( "option", "resizable", false );
```

showType: Boolean or Number or String or Object

Default: `null` dialog打开（显示）时的动画效果。支持多个类型：

- **Boolean**: 当设置为 `false`， 将不使用动画效果，该dialog会立即被隐藏。 如果设置为 `true`， 该dialog将会以默认的持续时间和默认的效果淡出。
- **Number**: 该dialog将以指定的时间和默认的效果淡出。
- **String**: 该dialog将使用指定的效果被隐藏。 该值可以是一个jQuery内置的动画方法的名称， 如 `"slideUp"`， 或一个 jQuery UI 效果的名称， 如 `"fold"` 。 在这两种情况下，将使用默认持续时间和默认的动画效果。
- **Object**: 如果该值是一个对象， 那么 `effect`, `delay`, `duration`, 和 `easing` 可能要提供。 如果 `effect` 属性包含一个jQuery方法的名称， 那么该方法将被使用; 否则它被假定为是一个jQuery UI的效果的名称。 当使用jQuery UI 支持额外设置 的效果， 你可以在对象中包含那些设置 并且它们将被传递到的效果。如果 `duration` 持续时间或 `easing` 属性被省略， 那么默认值将被使用。 如果 `effect` 被省略， 那么 `"fadeOut"` 将被使用。如果 `delay` 被省略， 那么将不使用延迟。

Code examples:

初始化带有指定 `show` 选项的 dialog：

```
$( ".selector" ).dialog({ show: { effect: "blind", duration: 800 } });
```

在初始化后，获取或设置 `show` 选项：

```
// getter
var show = $( ".selector" ).dialog( "option", "show" );

// setter
$( ".selector" ).dialog( "option", "show", { effect: "blind", duration: 800 } );
```

titleType: String

Default: `null` 指定dialog的标题文字。 如果值为 `null` ,那么该dialog 元素上的 `title` 属性将被使用。**Code examples:**

初始化带有指定 `title` 选项的 dialog：

```
$( ".selector" ).dialog({ title: "Dialog Title" });
```

在初始化后，获取或设置 `title` 选项：

```
// getter
var title = $( ".selector" ).dialog( "option", "title" );

// setter
$( ".selector" ).dialog( "option", "title", "Dialog Title" );
```

widthType: Number

Default: `300` 设置dialog的宽度（单位：像素）。**Code examples:**

初始化带有指定 `width` 选项的 dialog：

```
$( ".selector" ).dialog({ width: 500 });
```

在初始化后，获取或设置 `width` 选项：

```
// getter
var width = $( ".selector" ).dialog( "option", "width" );

// setter
$( ".selector" ).dialog( "option", "width", 500 );
```

Methods

close()Returns: jQuery (plugin only)

关闭dialog.

- 该方法不接受任何参数。

Code examples:

调用 close 方法：

```
$( ".selector" ).dialog( "close" );
```

destroy()Returns: jQuery (plugin only)

完全移除 dialog 功能. 这将使元素返回到之前的初始化状态.

- 该方法不接受任何参数。

Code examples:

调用 destroy 方法：

```
$( ".selector" ).dialog( "destroy" );
```

isOpen()Returns: Boolean

确定 dialog 当前是否打开状态。

- 该方法不接受任何参数。

Code examples:

调用 isOpen 方法：

```
var isOpen = $( ".selector" ).dialog( "isOpen" );
```

moveToTop()Returns: jQuery (plugin only)

移动dialog到所有dialog堆栈的顶部。（愚人码头注：理解为 z-index层级最高）

- 该方法不接受任何参数。

Code examples:

调用 moveToTop 方法：

```
$( ".selector" ).dialog( "moveToTop" );
```

open()Returns: jQuery (plugin only)

打开 dialog。

- 该方法不接受任何参数。

Code examples:

调用 open 方法：

```
$( ".selector" ).dialog( "open" );
```

option(optionName)Returns: Object

获取当前与指定的 `optionName` 关联的值。

- **optionName**Type: String要获取值的选项的名称。

Code examples:

调用该方法：

```
var isDisabled = $( ".selector" ).dialog( "option", "disabled" );
```

option()Returns: PlainObject

获取一个包含键/值对的对象，键/值对表示当前 dialog 选项哈希。

- 该方法不接受任何参数。

Code examples:

调用该方法：

```
var options = $( ".selector" ).dialog( "option" );
```

option(optionName, value)Returns: jQuery (plugin only)

设置与指定的 `optionName` 关联的 dialog 选项的值。

- **optionName**Type: String要设置的选项的名称。
- **value**Type: Object要为选项设置的值。

Code examples:

调用该方法：

```
$( ".selector" ).dialog( "option", "disabled", true );
```

option(options)Returns: jQuery (plugin only)

为 dialog 设置一个或多个选项。

- **options**Type: Object要设置的 option-value 对。

Code examples:

调用该方法：

```
$( ".selector" ).dialog( "option", { disabled: true } );
```

widget()Returns: jQuery

返回一个包含 生成包裹元素 的 `jQuery` 对象。

- 该方法不接受任何参数。

Code examples:

调用 widget 方法：

```
var widget = $( ".selector" ).dialog( "widget" );
```

扩展点（Extension Points）

dialog小部件是widget factory构建的，并且可以扩展。 当扩展部件时， 你必须覆盖或添加新的行为到现有的方法。 下列方法被提供作为扩展点 用相同的API稳定性如上所列的plugin methods。 有关小部件扩展的更多信息， 请参阅使用Widget Factory 扩展小部件。

_allowInteraction(event)Returns: Boolean

带遮罩的对话框不允许用户与对话框下面元素进行交互。 这可能会对 不属于该dialog的子元素那些绝对定位的其他元素产生问题。 `_allowInteraction()` 方法确定用户是否应被允许与给定的目标元素进行交互; 因此， 它可用于不属于该dialog的子元素白名单中的元素 但你希望用户能够使用。

- **event**Type: Event

Code examples:

允许Select2插件在遮罩对话框中使用。 `_super()` 调用确保该对话框中的元素仍然可以交互。

```
_allowInteraction: function( event ) {
  return !!$( event.target ).is( ".select2-input" ) || this._super( event );
}
```

Events

beforeClose(event, ui)Type: `dialogbeforeclose`

当dialog即将关闭时触发。 如果取消，dialog将不会关闭。

- **event**Type: Event
- **ui**Type: Object

注意： `ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

Code examples:

初始化带有指定 beforeClose 回调的 dialog：

```
$( ".selector" ).dialog({
  beforeClose: function( event, ui ) {}
});
```

绑定一个事件监听器到 dialogbeforeclose 事件：

```
$( ".selector" ).on( "dialogbeforeclose", function( event, ui ) {} );
```

close(event, ui)Type: `dialogclose`

当dialog关闭时触发。

- **event**Type: Event
- **ui**Type: Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

Code examples:

初始化带有指定 close 回调的 dialog：

```
$( ".selector" ).dialog({
  close: function( event, ui ) {}
});
```

绑定一个事件监听器到 dialogclose 事件：

```
$( ".selector" ).on( "dialogclose", function( event, ui ) {} );
```

create(event, ui)Type: `dialogcreate`

在创建dialog时触发。

- **event**Type: Event
- **ui**Type: Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

Code examples:

初始化带有指定 create 回调的 dialog：

```
$( ".selector" ).dialog({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 dialogcreate 事件：

```
$( ".selector" ).on( "dialogcreate", function( event, ui ) {} );
```

drag(event, ui)Type: `dialogdrag`

在dialog正在被拖动时触发。

- **event**Type: Event
- **ui**Type: Object
    - **position**Type: Objectdialog当前的CSS position（位置）对象。
    - **offset**Type: Objectdialog当前的offset position（偏移位置）对象。

Code examples:

初始化带有指定 drag 回调的 dialog：

```
$( ".selector" ).dialog({
  drag: function( event, ui ) {}
});
```

绑定一个事件监听器到 dialogdrag 事件：

```
$( ".selector" ).on( "dialogdrag", function( event, ui ) {} );
```

dragStart(event, ui)Type: `dialogdragstart`

当用户开始拖动dialog时触发。

- **event**Type: Event
- **ui**Type: Object
    - **position**Type: Objectdialog当前的CSS position（位置）对象。
    - **offset**Type: Objectdialog当前的offset position（偏移位置）对象。

Code examples:

初始化带有指定 dragStart 回调的 dialog：

```
$( ".selector" ).dialog({
  dragStart: function( event, ui ) {}
});
```

绑定一个事件监听器到 dialogdragstart 事件：

```
$( ".selector" ).on( "dialogdragstart", function( event, ui ) {} );
```

dragStop(event, ui)Type: `dialogdragstop`

当dialog 停止拖动时触发。

- **event**Type: Event
- **ui**Type: Object
  - **position**Type: Objectdialog当前的CSS position（位置）对象。
  - **offset**Type: Objectdialog当前的offset position（偏移位置）对象。

Code examples:

初始化带有指定 dragStop 回调的 dialog：

```
$( ".selector" ).dialog({
  dragStop: function( event, ui ) {}
});
```

绑定一个事件监听器到 dialogdragstop 事件：

```
$( ".selector" ).on( "dialogdragstop", function( event, ui ) {} );
```

focus(event, ui)Type: `dialogfocus`

当对话框获取焦点时触发此事件。

- **event**Type: Event
- **ui**Type: Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

Code examples:

初始化带有指定 focus 回调的 dialog：

```
$( ".selector" ).dialog({
  focus: function( event, ui ) {}
});
```

绑定一个事件监听器到 dialogfocus 事件：

```
$( ".selector" ).on( "dialogfocus", function( event, ui ) {} );
```

open(event, ui)Type: `dialogopen`

当对话框打开后，触发此事件。

- **event**Type: Event
- **ui**Type: Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

Code examples:

初始化带有指定 open 回调的 dialog：

```
$( ".selector" ).dialog({
  open: function( event, ui ) {}
});
```

绑定一个事件监听器到 dialogopen 事件：

```
$( ".selector" ).on( "dialogopen", function( event, ui ) {} );
```

resize(event, ui)Type: `dialogresize`

当对话框大小改变时，触发此事件。

- **event**Type: Event
- **ui**Type: Object
  - **originalPosition**Type: Object对话框被调整大小 之前的CSS position（位置）对象。
  - **position**Type: Object对话框当前的CSS position（位置）对象。
  - **originalSize**Type: Object对话框被调整大小 之前的size（尺寸）对象 。
  - **size**Type: Object对话框当前的size（尺寸）对象。

Code examples:

初始化带有指定 resize 回调的 dialog：

```
$( ".selector" ).dialog({
  resize: function( event, ui ) {}
});
```

绑定一个事件监听器到 dialogresize 事件：

```
$( ".selector" ).on( "dialogresize", function( event, ui ) {} );
```

resizeStart(event, ui)Type: `dialogresizestart`

当开始改变对话框大小时，触发此事件。

- **event**Type: Event
- **ui**Type: Object
  - **originalPosition**Type: Object对话框被调整大小 之前的CSS position（位置）对象。
  - **position**Type: Object对话框当前的CSS position（位置）对象。
  - **originalSize**Type: Object对话框被调整大小 之前的size（尺寸）对象 。
  - **size**Type: Object对话框当前的size（尺寸）对象。

Code examples:

初始化带有指定 resizeStart 回调的 dialog：

```
$( ".selector" ).dialog({
  resizeStart: function( event, ui ) {}
});
```

绑定一个事件监听器到 dialogresizestart 事件：

```
$( ".selector" ).on( "dialogresizestart", function( event, ui ) {} );
```

resizeStop(event, ui)Type: `dialogresizestop`

当对话框改变大小后，触发此事件。

- **event**Type: Event
- **ui**Type: Object
  - **originalPosition**Type: Object对话框被调整大小 之前的CSS position（位置）对象。
  - **position**Type: Object对话框当前的CSS position（位置）对象。
  - **originalSize**Type: Object对话框被调整大小 之前的size（尺寸）对象 。
  - **size**Type: Object对话框当前的size（尺寸）对象。

Code examples:

初始化带有指定 resizeStop 回调的 dialog：

```
$( ".selector" ).dialog({
  resizeStop: function( event, ui ) {}
});
```

绑定一个事件监听器到 dialogresizestop 事件：

```
$( ".selector" ).on( "dialogresizestop", function( event, ui ) {} );
```

Example:

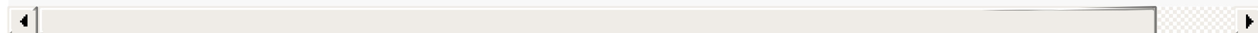
一个简单的 **jQuery UI** 对话框（**Dialog**）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>dialog demo</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-ui.cs
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<button id="opener">open the dialog</button>
<div id="dialog" title="Dialog Title">I'm a dialog</div>

<script>
$( "#dialog" ).dialog({ autoOpen: false });
$( "#opener" ).click(function() {
  $( "#dialog" ).dialog( "open" );
});
</script>

</body>
</html>
```



Menu Widget

Categories: Widgets

version added: 1.9

Description: 带有鼠标和键盘交互的用于导航的可主题化菜单。

QuickNavExamples

Options

- disabled
- icons
- menus
- position
- role

Methods

- blur
- collapse
- collapseAll
- destroy
- disable
- enable
- expand
- focus
- isFirstItem
- isLastItem
- next
- nextPage
- option
- previous
- previousPage
- refresh
- select
- widget

Events

- blur
- create
- focus
- select

菜单可以用任何有效的标记创建，只要元素有严格的父/子关系且每个条目都有一个锚。最常用的元素是无序列表（`<ul>`）：

```
<ul id="menu">
  <li><a href="#">Item 1</a></li>
  <li><a href="#">Item 2</a></li>
  <li><a href="#">Item 3</a>
    <ul>
      <li><a href="#">Item 3-1</a></li>
      <li><a href="#">Item 3-2</a></li>
      <li><a href="#">Item 3-3</a></li>
      <li><a href="#">Item 3-4</a></li>
      <li><a href="#">Item 3-5</a></li>
    </ul>
  </li>
  <li><a href="#">Item 4</a></li>
  <li><a href="#">Item 5</a></li>
</ul>
```

如果使用一个非 `<ul>` / `<li>` 的结构，为菜单和菜单条目使用相同的元素，请使用 `menus` 选项来区分两个元素，例如 `menus: "div.menuElement"`。

可通过向元素添加 `ui-state-disabled` class 来禁用任何菜单条目。

图标（Icons）

为了向菜单添加图标，请在标记中包含图标：

```
<ul id="menu">
  <li><a href="#"><span class="ui-icon ui-icon-disk"></span>Save</a></li>
</ul>
```

菜单（Menu）会自动向无图标的条目添加必要的内边距。

分隔符（Dividers）

分隔符元素可通过包含未链接的菜单条目来创建，菜单条目只能是空格/破折号：

```
<ul id="menu">
  <li><a href="#">Item 1</a></li>
  <li>-</li>
  <li><a href="#">Item 2</a></li>
</ul>
```

键盘交互（Keyboard interaction）

- ENTER/SPACE：调用获得焦点的菜单项的动作，可能会打开一个子菜单。
- UP：移动教导到上一个菜单项。
- DOWN：移动教导到下一个菜单项。
- RIGHT：如果可用，则打开子菜单。
- LEFT：关闭当前子菜单，移动焦点到父菜单项。如果焦点不在子菜单上，则不进行任何操作。
- ESCAPE：关闭当前子菜单，移动焦点到父菜单项。如果焦点不在子菜单上，则不进行任何操作。

输入一个字母，移动焦点到以该字母开头的第一个条目。重复相同的字符会循环显示匹配的条目。在一个时间内输入更多的字符则匹配所输入的字符。

禁用项可获得键盘焦点，但是不允许任何交互。

主题化（Theming）

菜单部件（Menu Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用菜单指定的样式，则可以使用下面的 CSS class 名称：

- `ui-menu`：菜单的外层容器。如果菜单包含图标，该元素会另外带有一个 `ui-menu-icons` class。
  - `ui-menu-item`：单个菜单项的容器。
    - `ui-menu-icon`：通过 `icons` 选项进行子菜单图标设置。
  - `ui-menu-divider`：菜单项之间的分隔符元素。

依赖（Dependencies）

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 定位（Position）

其他注意事项（Additional Notes:）

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

Options

disabledType: Boolean

Default: `false` 如果设置为 `true`，则禁用该 menu（菜单）。**Code examples:**

初始化带有指定 `disabled` 选项的menu（菜单）

```
$( ".selector" ).menu({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).menu( "option", "disabled" );

// setter
$( ".selector" ).menu( "option", "disabled", true );
```

iconsType: Object

Default: `{ submenu: "ui-icon-carat-1-e" }` 标题要使用的图标，与 jQuery UI CSS 框架提供的图标（Icons） 匹配。设置为 false 则不显示图标。

- submenu (string, default: "ui-icon-carat-1-e")

Code examples:

初始化带有指定 `icons` 选项的menu（菜单）

```
$( ".selector" ).menu({ icons: { submenu: "ui-icon-circle-triangle-e" } });
```

在初始化后，获取或设置 `icons` 选项：

```
// getter
var icons = $( ".selector" ).menu( "option", "icons" );

// setter
$( ".selector" ).menu( "option", "icons", { submenu: "ui-icon-circle-triangle-e" } );
```

menusType: String

Default: `"ul"`

作为menu（菜单）容器的元素的选择器， 包括子菜单。

注意: 初始化后 `menus` 选项不应该被更改。 现有的子菜单将不会被更新。**Code examples:**

初始化带有指定 `menus` 选项的menu（菜单）

```
$( ".selector" ).menu({ menus: "div" });
```

获取 `menus` 选项：

```
// getter
var menus = $( ".selector" ).menu( "option", "menus" );
```

positionType: Object

Default: `{ my: "left top", at: "right top" }` 标识建议菜单的位置与相关的 input 元素有关系。 `of` 选项默认为 input 元素，但是您可以指定另一个定位元素。如需了解各种选项的更多细节，请查看 jQuery UI Position。**Code examples:**

初始化带有指定 `position` 选项的menu（菜单）

```
$( ".selector" ).menu({ position: { my: "left top", at: "right-5 top+5" } });
```

在初始化后，获取或设置 `position` 选项：

```
// getter
var position = $( ".selector" ).menu( "option", "position" );

// setter
$( ".selector" ).menu( "option", "position", { my: "left top", at: "right-5 top+5" } );
```

roleType: String

Default: `"menu"`

自定义用于菜单和菜单项的ARIA roles（愚人码头注：关于ARIA roles）。 在默认情况下菜单项使用 `"menuitem"` 。 设置 `role` 选项为了使 `"listbox"` 使用 `"option"` 作为菜单项。 如果设置为 `null` ， 没有roles将被设置，如果菜单是由被保持焦点另一个元件控制时候，非常有用。

注意: 初始化后 `role` 选项 不应该被更改。 现有（子）菜单和菜单项将不会被更新。**Code examples:**

初始化带有指定 `role` 选项的menu（菜单）

```
$( ".selector" ).menu({ role: null });
```

获取 `role` 选项：

```
// getter
var role = $( ".selector" ).menu( "option", "role" );
```

Methods

blur([event])Returns: jQuery (plugin only)

从菜单中删除焦点， 重置任何激活样式 和 触发菜单的 `blur` 事件。

- **event**Type: Event什么事件触发了菜单失去焦点。

Code examples:

调用 blur 方法：

```
$( ".selector" ).menu( "blur" );
```

collapse([event])Returns: jQuery (plugin only)

关闭当前活动的子菜单。

- **event**Type: Event什么事件触发关闭子菜单

Code examples:

调用 collapse 方法：

```
$( ".selector" ).menu( "collapse" );
```

collapseAll([event] [, all])Returns: jQuery (plugin only)

关闭全部打开的子菜单。

- **event**Type: Event什么事件触发关闭子菜单。
- **all**Type: Boolean表示所有子菜单是否应该被关闭 或 只有子菜单中包括的菜单 或 包含触发事件的目标元素。

Code examples:

调用 collapseAll 方法：

```
$( ".selector" ).menu( "collapseAll", null, true );
```

destroy()Returns: jQuery (plugin only)

完全移除 menu 功能. 这将使元素返回到之前的初始化状态.

- 该方法不接受任何参数。

Code examples:

调用 destroy 方法：

```
$( ".selector" ).menu( "destroy" );
```

disable()Returns: jQuery (plugin only)

禁用 menu.

- 该方法不接受任何参数。

Code examples:

调用 disable 方法：

```
$( ".selector" ).menu( "disable" );
```

enable()Returns: jQuery (plugin only)

启用 menu.

- 该方法不接受任何参数。

Code examples:

调用 enable 方法：

```
$( ".selector" ).menu( "enable" );
```

expand([event])Returns: jQuery (plugin only)

打开当前活动项目下的子菜单下，如果有的话。

- **event**Type: Event什么时间触发打开子菜单。

Code examples:

调用 expand 方法：

```
$( ".selector" ).menu( "expand" );
```

focus([event], item)Returns: jQuery (plugin only)

激活一个特定的菜单项， 首先，如果打开存在的任何子菜单，并触发菜单的 `focus` 事件。

- **event**Type: Event什么事件触发了菜单项获得焦点。
- **item**Type: jQuery要获取焦点/激活的菜单项

Code examples:

调用 focus 方法：

```
$( ".selector" ).menu( "focus", null, menu.find( ".ui-menu-item:last" ) );
```

isFirstItem()Returns: jQuery (plugin only)

返回一个布尔值，说明当前活动项目是否菜单的第一项。

- 该方法不接受任何参数。

Code examples:

调用 isFirstItem 方法：

```
var firstItem = $( ".selector" ).menu( "isFirstItem" );
```

isLastItem()Returns: jQuery (plugin only)

返回一个布尔值，说明当前活动项目是否菜单的最后一项。

- 该方法不接受任何参数。

Code examples:

调用 isLastItem 方法：

```
var lastItem = $( ".selector" ).menu( "isLastItem" );
```

next([event])Returns: jQuery (plugin only)

移动激活状态到下一个菜单项。

- **event**Type: Event什么事件触发焦点转移。

Code examples:

调用 next 方法：

```
$( ".selector" ).menu( "next" );
```

nextPage([event])Returns: jQuery (plugin only)

移动激活状态到第一个菜单项下的可滚动菜单的底部，或最后一个项目，如果不可滚动。

- **event**Type: Event什么事件触发焦点转移。

Code examples:

调用 nextPage 方法：

```
$( ".selector" ).menu( "nextPage" );
```

option(optionName)Returns: Object

获取当前与指定的 `optionName` 关联的值。

- **optionName**Type: String要获取值的选项的名称。

Code examples:

调用该方法：

```
var isDisabled = $( ".selector" ).menu( "option", "disabled" );
```

option()Returns: PlainObject

获取一个包含键/值对的对象，键/值对表示当前 menu 选项哈希。

- 该方法不接受任何参数。

Code examples:

调用该方法：

```
var options = $( ".selector" ).menu( "option" );
```

option(optionName, value)Returns: jQuery (plugin only)

设置与指定的 `optionName` 关联的 menu 选项的值。

- **optionName**Type: String要设置的选项的名称。
- **value**Type: Object要为选项设置的值。

Code examples:

调用该方法：

```
$( ".selector" ).menu( "option", "disabled", true );
```

option(options)Returns: jQuery (plugin only)

为 menu 设置一个或多个选项。

- **options**Type: Object要设置的 option-value 对。

Code examples:

调用该方法：

```
$( ".selector" ).menu( "option", { disabled: true } );
```

previous([event])Returns: jQuery (plugin only)

移动激活状态到上一个菜单项。

- **event**Type: Event什么事件触发焦点转移。

Code examples:

调用 previous 方法：

```
$( ".selector" ).menu( "previous" );
```

previousPage([event])Returns: jQuery (plugin only)

移动激活状态到第一个菜单项下的可滚动菜单的顶部，或第一一个项目，如果不可滚动。

- **event**Type: Event什么事件触发焦点转移。

Code examples:

调用 previousPage 方法：

```
$( ".selector" ).menu( "previousPage" );
```

refresh()Returns: jQuery (plugin only)

初始化还没有被初始化的子菜单和菜单项。 新的菜单项， 包括子菜单可以被添加到菜单 或所有的菜单的内容可以被替换 然后使用 `refresh()` 方法初始化。

- 该方法不接受任何参数。

Code examples:

调用 refresh 方法：

```
$( ".selector" ).menu( "refresh" );
```

select([event])Returns: jQuery (plugin only)

选择当前活动的菜单项，折叠所有子菜单 并触发菜单中的 `select` 事件。

- **event**Type: Event什么事件触发选择。

Code examples:

调用 select 方法：

```
$( ".selector" ).menu( "select" );
```

widget()Returns: jQuery

返回一个包含 button 的 `jQuery` 对象。

- 该方法不接受任何参数。

Code examples:

调用 widget 方法：

```
var widget = $( ".selector" ).menu( "widget" );
```

Events

blur(event, ui)Type: `menublur`

当menu失去焦点时触发这个事件。

- **event**Type: Event
- **ui**Type: Object
  - **item**Type: jQuery当前激活的菜单项。

Code examples:

初始化带有指定 blur 回调的 menu：

```
$( ".selector" ).menu({
  blur: function( event, ui ) {}
});
```

绑定一个事件监听器到 menublur 事件：

```
$( ".selector" ).on( "menublur", function( event, ui ) {} );
```

create(event, ui)Type: `menucreate`

当创建 menu 时触发。

- **event**Type: Event
- **ui**Type: Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

Code examples:

初始化带有指定 create 回调的 menu：

```
$( ".selector" ).menu({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 menucreate 事件：

```
$( ".selector" ).on( "menucreate", function( event, ui ) {} );
```

focus(event, ui)Type: `menufocus`

当当一个菜单获得焦点或当任意一个菜单项被激活时触发这个事件。

- **event**Type: Event
- **ui**Type: Object
    - **item**Type: jQuery当前激活的菜单项。

Code examples:

初始化带有指定 focus 回调的 menu：

```
$( ".selector" ).menu({
  focus: function( event, ui ) {}
});
```

绑定一个事件监听器到 menufocus 事件：

```
$( ".selector" ).on( "menufocus", function( event, ui ) {} );
```

select(event, ui)Type: `menuselect`

当才安被选中的时候触发该事件。

- **event**Type: Event
- **ui**Type: Object
  - **item**Type: jQuery当前激活的菜单项。

Code examples:

初始化带有指定 select 回调的 menu：

```
$( ".selector" ).menu({
  select: function( event, ui ) {}
});
```

绑定一个事件监听器到 menuselect 事件：

```
$( ".selector" ).on( "menuselect", function( event, ui ) {} );
```

Example:

一个简单的 jQuery UI Menu

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>menu demo</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-ui.cs
  <style>
  .ui-menu {
    width: 200px;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<ul id="menu">
  <li><a href="#">Item 1</a></li>
  <li><a href="#">Item 2</a></li>
  <li><a href="#">Item 3</a>
    <ul>
      <li><a href="#">Item 3-1</a></li>
      <li><a href="#">Item 3-2</a></li>
      <li><a href="#">Item 3-3</a></li>
      <li><a href="#">Item 3-4</a></li>
      <li><a href="#">Item 3-5</a></li>
    </ul>
  </li>
  <li><a href="#">Item 4</a></li>
  <li><a href="#">Item 5</a></li>
</ul>

<script>
$( "#menu" ).menu();
</script>

</body>
</html>
```

Progressbar Widget

Categories: Widgets

version added: 1.6

Description: 显示一个确定的或不确定的进程状态。

QuickNavExamples

Options

- disabled
- max
- value

Methods

- destroy
- disable
- enable
- option
- value
- widget

Events

- change
- complete
- create

进度条被设计来显示进度的当前完成百分比。进度条通过 CSS 编码灵活调整大小，默认会缩放到适应父容器的大小。

一个确定的进度条只能在系统可以准确更新当前状态的情况下使用。一个确定的进度条不会从左向右填充，然后循环回到空 - 如果不能计算实际状态，则使用不确定的进度条以便提供用户反馈。

主题（Theming）

进度条部件（Progressbar Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用进度条指定的样式，则可以使用下面的 CSS class 名称：

- `ui-progressbar`：进度条的外层容器。该元素会为不确定的进度条另外添加一个 `ui-progressbar-indeterminate` class。
    - `ui-progressbar-value`：该元素代表进度条的填充部分。
        - `ui-progressbar-overlay`：用于为不确定的进度条显示动画的覆盖层。

依赖（Dependencies）

- UI 核心（UI Core）
- 部件库（Widget Factory）

其他注意事项（Additional Notes）：

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

Options

disabledType: Boolean

Default: `false` 如果设置为 `true`，则禁用该 progressbar（进度条）。 **Code examples:**

初始化带有指定 `disabled` 选项的 progressbar（进度条）：

```
$( ".selector" ).progressbar({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).progressbar( "option", "disabled" );

// setter
$( ".selector" ).progressbar( "option", "disabled", true );
```

maxType: Number

Default: `100` progressbar（进度条）的最大值。**Code examples:**

初始化带有指定 `max` 选项的 progressbar（进度条）：

```
$( ".selector" ).progressbar({ max: 1024 });
```

在初始化后，获取或设置 `max` 选项：

```
// getter
var max = $( ".selector" ).progressbar( "option", "max" );

// setter
$( ".selector" ).progressbar( "option", "max", 1024 );
```

valueType: Number or Boolean

Default: `0` progressbar（进度条）的进度值.支持多个类型：

- **Number**: `0` 到 `max` 之间的值.
- **Boolean**:值可以设置为 `false` 来创建一个不确定的progressbar（进度条）。

Code examples:

初始化带有指定 `value` 选项的 progressbar（进度条）：

```
$( ".selector" ).progressbar({ value: 25 });
```

在初始化后，获取或设置 `value` 选项：

```
// getter
var value = $( ".selector" ).progressbar( "option", "value" );

// setter
$( ".selector" ).progressbar( "option", "value", 25 );
```

Methods

destroy()Returns: jQuery (plugin only)

完全移除 progressbar（进度条） 功能。这会把元素返回到它的预初始化状态。

- 该方法不接受任何参数。

Code examples:

调用 destroy 方法：

```
$( ".selector" ).progressbar( "destroy" );
```

disable()Returns: jQuery (plugin only)

禁用 progressbar（进度条）。

- 该方法不接受任何参数。

Code examples:

调用 disable 方法：

```
$( ".selector" ).progressbar( "disable" );
```

enable()Returns: jQuery (plugin only)

启用 progressbar（进度条）。

- 该方法不接受任何参数。

Code examples:

调用 enable 方法：

```
$( ".selector" ).progressbar( "enable" );
```

option(optionName)Returns: Object

获取当前与指定的 `optionName` 关联的值。

- **optionName**Type: String要获取值的选项的名称。

Code examples:

调用该方法：

```
var isDisabled = $( ".selector" ).progressbar( "option", "disabled" );
```

option()Returns: PlainObject

获取一个包含键/值对的对象，键/值对表示当前 progressbar 选项哈希。

- 该方法不接受任何参数。

Code examples:

调用该方法：

```
var options = $( ".selector" ).progressbar( "option" );
```

option(optionName, value)Returns: jQuery (plugin only)

设置与指定的 `optionName` 关联的 progressbar 选项的值。

- **optionName**Type: String要设置的选项的名称。
- **value**Type: Object要为选项设置的值。

Code examples:

调用该方法：

```
$( ".selector" ).progressbar( "option", "disabled", true );
```

option(options)Returns: jQuery (plugin only)

为 progressbar（进度条） 设置一个或多个选项。

- **options**Type: Object要设置的 option-value 对。

Code examples:

调用该方法：

```
$( ".selector" ).progressbar( "option", { disabled: true } );
```

value()Returns: Number or Boolean

获取progressbar（进度条）的当前值。

- 该方法不接受任何参数。

Code examples:

调用该方法：

```
var progressSoFar = $( ".selector" ).progressbar( "value" );
```

value(value)Returns: jQuery (plugin only)

设置progressbar（进度条）的当前值。

- **value**Type: Number or Boolean用来设置的值。有效值的详细描述查看 `value` 选项。

Code examples:

调用该方法：

```
$( ".selector" ).progressbar( "value", 50 );
```

widget()Returns: jQuery

返回一个 progressbar（进度条） 的 `jQuery` 对象。

- 该方法不接受任何参数。

Code examples:

调用 widget 方法：

```
var widget = $( ".selector" ).progressbar( "widget" );
```

Events

change(event, ui)Type: `progressbarchange`

当 progressbar（进度条） 的值改变的时候触发该事件。

- **event**Type: Event
- **ui**Type: Object

注意： *`ui`* 对象是空的，这里包含它是为了与其他事件保持一致性。

Code examples:

初始化带有指定 change回调的 progressbar（进度条）：

```
$( ".selector" ).progressbar({
  change: function( event, ui ) {}
});
```

绑定一个事件监听器到 progressbarchange 事件：

```
$( ".selector" ).on( "progressbarchange", function( event, ui ) {} );
```

complete(event, ui)Type: `progressbarcomplete`

当progressbar（进度条）达到最大值时触发该事件。

- **event**Type: Event
- **ui**Type: Object

注意： *`ui`* 对象是空的，这里包含它是为了与其他事件保持一致性。

Code examples:

初始化带有指定complete回调的 progressbar（进度条）：

```
$( ".selector" ).progressbar({
  complete: function( event, ui ) {}
});
```

绑定一个事件监听器到 progressbarcomplete 事件：

```
$( ".selector" ).on( "progressbarcomplete", function( event, ui ) {} );
```

create(event, ui)Type: `progressbarcreate`

当progressbar（进度条）被创建时触发该事件。

- **event**Type: Event
- **ui**Type: Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

Code examples:

初始化带有指定create回调的 progressbar（进度条）：

```
$( ".selector" ).progressbar({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 progressbarcreate 事件：

```
$( ".selector" ).on( "progressbarcreate", function( event, ui ) {} );
```

Examples:

Example: 一个简单的 jQuery UI Progressbar

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>progressbar demo</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-ui.cs
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="progressbar"></div>

<script>
$( "#progressbar" ).progressbar({
  value: 37
});
</script>

</body>
</html>
```

Example: 一个简单的 jQuery UI 不确定的进度条（Indeterminate Progressbar）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>progressbar demo</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-ui.cs
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="progressbar"></div>

<script>
$( "#progressbar" ).progressbar({
  value: false
});
</script>

</body>
</html>
```

Slider Widget

Categories: Widgets

version added: 1.5

Description: 拖动手柄以选择一个数值。

QuickNavExamples

Options

- animate
- disabled
- max
- min
- orientation
- range
- step
- value
- values

Methods

- destroy
- disable
- enable
- option
- value
- values
- widget

Events

- change
- create
- slide
- start

- stop

jQuery UI Slider 插件使所选择的元素成为一个滑块(slider). 可以多种选项,例如多个操作手柄和操作范围。 手柄可以被鼠标拖动或者随着方向键移动。

slider小工具将在初始化的时候根据 `ui-slider-handle` 样式名创建手柄元素。您可以在初始化前通过创建和追加的方式，或者在元素上添加 `ui-slider-handle` 样式来自定义手柄元素。 这只会创建需要匹配 `value` / `values` 的手柄数值个数。例如，如果您指定的 `值: [ 1, 5, 18 ]`，并创建一个自定义手柄，该插件将创建另外两个。

主题(Theming)

slider小工具使用jQuery UI CSS framework样式的外观和感觉。如果滑块具体样式是必须的，你可以用下面的CSS类名：

- `ui-slider`：滑块控件的轨道。该元素将追加一个 `ui-slider-horizontal` 或 `ui-slider-vertical` 样式名，这取决于slider小工具是横向还是纵向的。另外具有UI的滑块水平或UI滑块垂直取决于 `orientation` 参数，表示滑块的取向的类名。
    - `ui-slider-handle`：滑块手柄.
    - `ui-slider-range`：当设置 选项时使用的已选范围。如果 `range` 选项设置为 `"min"` 或者 `"max"`，则该元素会分别另外带有一个 `ui-slider-range-min` 或 `ui-slider-range-max` 类。

依赖

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 鼠标交互（Mouse Interaction）

其他注意事项:

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

选项

animate类型: Boolean or String or Number

默认值: `false` 当用户单击滑块轨道时是否平稳地滑动手柄。 也可以接受任何有效的动画持续时间。多种类型支持：

- **Boolean**: 当设置为 `true` 时, 滑动手柄将以默认的持续时间执行动画。

- **String**: 速度的名称, 比如 `"fast"` 或 `"slow"` 。
- **Number**: 动画持续时间, 以毫秒为单位。

代码示例:

使用指定 `animate` 选项初始化滑块：

```
$( ".selector" ).slider({ animate: "fast" });
```

初始化后，获取或者设置 `animate` 选项:

```
// getter
var animate = $( ".selector" ).slider( "option", "animate" );

// setter
$( ".selector" ).slider( "option", "animate", "fast" );
```

disabledType: Boolean

Default: `false` 如果设置为 `true` ，则禁用滑块。**Code examples:**

使用 `disabled` 选项初始化滑块组件：

```
$( ".selector" ).slider({ disabled: true });
```

在组件初始化之后，读取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).slider( "option", "disabled" );

// setter
$( ".selector" ).slider( "option", "disabled", true );
```

maxType: Number

Default: `100` 滑块的最大值。**Code examples:**

使用 `max` 选项初始化滑块组件：

```
$( ".selector" ).slider({ max: 50 });
```

在组件初始化之后，读取或设置 `max` 选项：

```
// getter
var max = $( ".selector" ).slider( "option", "max" );

// setter
$( ".selector" ).slider( "option", "max", 50 );
```

minType: Number

Default: `0` 滑块的最小值**Code examples:**

使用 `min` 选项初始化滑块组件：

```
$( ".selector" ).slider({ min: 10 });
```

在组件初始化之后，读取或设置 `min` 选项：

```
// getter
var min = $( ".selector" ).slider( "option", "min" );

// setter
$( ".selector" ).slider( "option", "min", 10 );
```

orientationType: String

Default: `"horizontal"` 确定滑块手柄 将 水平（最小在左，最大在右）或垂直（最小在底部，最大在顶部）移动。（愚人码头注：就是滑块的方向，横向或纵向）可能的值：`"horizontal"`（横向），`"vertical"`（纵向）。**Code examples:**

使用 `orientation` 选项初始化滑块组件：

```
$( ".selector" ).slider({ orientation: "vertical" });
```

在组件初始化之后，读取或设置 `orientation` 选项：

```
// getter
var orientation = $( ".selector" ).slider( "option", "orientation" );

// setter
$( ".selector" ).slider( "option", "orientation", "vertical" );
```

rangeType: Boolean or String

Default: `false` 滑块是否表现为一个范围。多种类型支持：

- **Boolean**: 如果设置为 `true` ,如果你有两个手柄，slider将会感应到他们之间各种可能的范围要素。

- **String**: 或者他们是 `"min"` 或者 `"max"` 值。最小的范围将从slider的最小值传递给操作柄。最大的范围将从slider的最大值传递给操作柄

Code examples:

使用 `range` 选项初始化滑块组件：

```
$( ".selector" ).slider({ range: true });
```

在组件初始化之后，读取或设置 `range` 选项：

```
// getter
var range = $( ".selector" ).slider( "option", "range" );

// setter
$( ".selector" ).slider( "option", "range", true );
```

stepType: Number

Default: `1` 定义slider从最小值移动到最大值的单位步长。在这个指定范围(最小值到最大值)内的值需要能被范围整除。**Code examples:**

使用 `step` 选项初始化滑块组件：

```
$( ".selector" ).slider({ step: 5 });
```

在组件初始化之后，读取或设置 `step` 选项：

```
// getter
var step = $( ".selector" ).slider( "option", "step" );

// setter
$( ".selector" ).slider( "option", "step", 5 );
```

valueType: Number

Default: `0` 如果只有一个手柄则是指定slider的value值。 如果有多余一个的操作柄, 则是定义第一个操作柄的value值。**Code examples:**

使用 `value` 选项初始化滑块组件：

```
$( ".selector" ).slider({ value: 10 });
```

在组件初始化之后，读取或设置 `value` 选项：

```
// getter
var value = $( ".selector" ).slider( "option", "value" );

// setter
$( ".selector" ).slider( "option", "value", 10 );
```

valuesType: Array

Default: `null` 这个选项可以用来为多个操作柄设定value值. 如果 `range` 设置为 `true` , 'values' 的长度最少应为 2。**Code examples:**

使用 `values` 选项初始化滑块组件：

```
$( ".selector" ).slider({ values: [ 10, 25 ] });
```

在组件初始化之后，读取或设置 `values` 选项：

```
// getter
var values = $( ".selector" ).slider( "option", "values" );

// setter
$( ".selector" ).slider( "option", "values", [ 10, 25 ] );
```

Methods

destroy()Returns: jQuery (plugin only)

完全销毁滑块组件的功能，这将使元素返回它的初始状态。

- 这个方法不接收任何参数

Code examples:

调用destroy 方法：

```
$( ".selector" ).slider( "destroy" );
```

disable()Returns: jQuery (plugin only)

禁用滑块组件。

- 这个方法不接收任何参数

Code examples:

调用disable 方法：

```
$( ".selector" ).slider( "disable" );
```

enable()Returns: jQuery (plugin only)

启用滑块组件。

- 这个方法不接收任何参数

Code examples:

调用enable 方法：

```
$( ".selector" ).slider( "enable" );
```

option(optionName)Returns: Object

获取与 `optionName` 对应的选项值。

- **optionName**Type: String要获取的值所对应的选项的名称。

Code examples:

调用这个方法：

```
var isDisabled = $( ".selector" ).slider( "option", "disabled" );
```

option()Returns: PlainObject

获取一个包含有描述当前滑块组件选项哈希值的键/值对。

- 这个方法不接收任何参数

Code examples:

调用这个方法：

```
var options = $( ".selector" ).slider( "option" );
```

option(optionName, value)Returns: jQuery (plugin only)

设置滑块组件 `optionName` 参数指定的选项的值。

- **optionName**Type: String要设置的选项的名称。
- **value**Type: Object要为选项设置的参数值。

Code examples:

调用这个方法：

```
$( ".selector" ).slider( "option", "disabled", true );
```

option(options)Returns: jQuery (plugin only)

设置一个或多个滑块组件的选项值。

- **options**Type: Object要设置的选项名与值之间的映射关系。

Code examples:

调用这个方法：

```
$( ".selector" ).slider( "option", { disabled: true } );
```

value()Returns: Number

获取滑块组件的值。

- 这个方法不接收任何参数

Code examples:

调用这个方法：

```
var selection = $( ".selector" ).slider( "value" );
```

value(value)Returns: jQuery (plugin only)

设置滑块组件的值。

- **value**Type: Number用来设置的值

Code examples:

调用这个方法：

```
$( ".selector" ).slider( "value", 55 );
```

values()Returns: Array

获取所有手柄的值.（愚人码头注：适用于多手柄的滑块）

- 这个方法不接收任何参数

Code examples:

调用这个方法：

```
var values = $( ".selector" ).slider( "values" );
```

values(index)Returns: Number

获取指定手柄的值。（愚人码头注：适用于多手柄的滑块）

- **index**Type: Integer从0开始计数的手柄索引值。

Code examples:

调用这个方法：

```
var value = $( ".selector" ).slider( "values", 0 );
```

values(index, value)Returns: jQuery (plugin only)

设置指定手柄的值。（愚人码头注：适用于多手柄的滑块）

- **index**Type: Integer从0开始计数的手柄索引值。
- **value**Type: Number要设置的值

Code examples:

调用这个方法：

```
$( ".selector" ).slider( "values", 0, 55 );
```

values(values)Returns: jQuery (plugin only)

设置所有手柄的值。（愚人码头注：适用于多手柄的滑块）

- **values**Type: Array要设置的值。

Code examples:

调用这个方法：

```
$( ".selector" ).slider( "values", [ 55, 105 ] );
```

widget()Returns: jQuery

返回一个滑块元素的 `jQuery` 对象。

- 这个方法不接收任何参数

Code examples:

调用widget 方法：

```
var widget = $( ".selector" ).slider( "widget" );
```

Events

change(event, ui)Type: `slidechange`

当用户滑动一个手柄后，如果滑块的值改变了，就会触发这个事件；或者通过 `value` 方法改变手柄值。

- **event**Type: Event
- **ui**Type: Object
  - **handle**Type: jQuery一个jQuery对象，表示被改变的手柄
  - **value**Type: Number当前滑块的值。

Code examples:

使用change 回调函数指定事件:

```
$( ".selector" ).slider({
  change: function( event, ui ) {}
});
```

绑定事件侦听器到slidechange事件：

```
$( ".selector" ).on( "slidechange", function( event, ui ) {} );
```

create(event, ui)Type: `slidecreate`

当滑块组件被创建时触发。

- **event**Type: Event
- **ui**Type: Object

注意: *`ui`* 对象是空的，但是为了与其它元素保持一直，它总是被包含。

Code examples:

使用create 回调函数指定事件:

```
$( ".selector" ).slider({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 slidecreate 事件:

```
$( ".selector" ).on( "slidecreate", function( event, ui ) {} );
```

slide(event, ui)Type: `slide`

这个事件鼠标滑动滑块时触发。 `ui.value` 作为提供给事件的价值，表示将作为该当前移动手柄的结果值。 取消该事件会阻止手柄移动和手柄将继续其先前的值。取消该事件会阻止手柄移动并且手柄将继续其先前的值。

- **event**Type: Event
- **ui**Type: Object
  - **handle**Type: jQuery一个jQuery对象，表示当前移动的手柄
  - **value**Type: Number如果事件没有被取消，该手柄将移动到值 。
  - **values**Type: Array一个 多手柄滑块 当前值的数组。

Code examples:

使用slide 回调函数指定事件:

```
$( ".selector" ).slider({
  slide: function( event, ui ) {}
});
```

绑定一个事件监听器到 slide 事件:

```
$( ".selector" ).on( "slide", function( event, ui ) {} );
```

start(event, ui)Type: `slidestart`

当用户开始滑动滑块时触发。

- **event**Type: Event
- **ui**Type: Object
  - **handle**Type: jQuery一个jQuery对象，表示当前移动的手柄
  - **value**Type: Number滑块的当前值

Code examples:

使用start 回调函数指定事件:

```
$( ".selector" ).slider({
  start: function( event, ui ) {}
});
```

绑定一个事件监听器到 slidestart 事件:

```
$( ".selector" ).on( "slidestart", function( event, ui ) {} );
```

stop(event, ui)Type: `slidestop`

当用户滑动滑块后触发。

- **event**Type: Event
- **ui**Type: Object
  - **handle**Type: jQuery一个jQuery对象，表示移动后手柄。
  - **value**Type: Number滑块的当前值

Code examples:

使用stop 回调函数指定事件:

```
$( ".selector" ).slider({
  stop: function( event, ui ) {}
});
```

绑定一个事件监听器到 slidestop 事件:

```
$( ".selector" ).on( "slidestop", function( event, ui ) {} );
```

Example:

一个简单的 **jQuery UI** 滑块（**Slider**）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>slider demo</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-ui.cs
  <style>#slider { margin: 10px; } </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="slider"></div>

<script>
$( "#slider" ).slider();
</script>

</body>
</html>
```

Spinner Widget

Categories: Widgets

version added: 1.9

Description: 通过向上/向下按钮和箭头按键操作，增强文本输入框的数值输入功能。

QuickNavExamples

Options

- culture
- disabled
- icons
- incremental
- max
- min
- numberFormat
- page
- step

Methods

- destroy
- disable
- enable
- option
- pageDown
- pageUp
- stepDown
- stepUp
- value
- widget

Extension Points

- _buttonHtml

- _uiSpinnerHtml

Events

- change
- create
- spin
- start
- stop

spinner（微调组件），或数步进控件（number stepper widget），是用于处理各种数字输入的完美控件。它允许用户直接输入一个值，或通过键盘、鼠标、滚轮微调改变一个已有的值。当与全球化（Globalize）结合时，您甚至可以微调显示不同地区的货币和日期。

spinner（微调组件）使用两个按钮将文本输入覆盖为当前值的递增值和递减值。spinner（微调组件）增加了按键事件，以便可以用键盘完成相同的递增和递减。spinner（微调组件）代表 全球化（Globalize）的数字格式和解析。

键盘交互（Keyboard interaction）

- UP：对值增加一步。
- DOWN：对值减少一步。
- PAGE UP：对值增加一页。
- PAGE DOWN：对值减少一页。

用鼠标点击微调按钮后，焦点仍停留在文本域中。

当spinner（微调组件）不是只读（ `<input readonly>` ）时，用户可以输入值，这可能会产生无效的值（小于最小值，大于最大值，增减错配，非数字输入）。当增减时，不管通过编程方式还是微调按钮方式，值都会被强制为一个有效值（如需了解详情，请查看 `stepUp()` 和 `stepDown()` 的描述。

主题（Theming）

spinner（微调组件）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用 spinner（微调组件）指定的样式，则可以使用下面的 CSS class 名称：

- `ui-spinner`：spinner（微调组件）的外层容器。
  - `ui-spinner-input`：spinner（微调组件）实例化的 `<input>` 元素。
  - `ui-spinner-button`：用于递增或递减spinner（微调组件）值的按钮控件。向上按钮会另外带有一个 `ui-spinner-up` class，向下按钮会另外带有一个 `ui-spinner-down` class。

依赖（Dependencies）

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 按钮部件（Button Widget）
- 全球化（Globalize）（外部的，可选的；当与 `culture` 和 `numberFormat` 选项一起使用时）

Additional Notes:

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。
- 该部件以编程方式操作元素的值，因此当元素的值改变时不会触发原生的 `change` 事件。
- 不支持在 `<input type="number">` 上创建选择器，因为会造成与本地spinner（微调组件）的 UI 冲突。

Options

cultureType: String

Default: `null` 设置 `culture` 选项 用于解析和格式化值。 如果为 `null`，在 `Globalize` 下当前设置的culture将被使用， 可供的 `culture`，请查看Globalize 文档。 只有当 `numberFormat` 选项设置了，才会有关联。 需要引用Globalize。**Code examples:**

初始化带有指定 `culture` 选项的 spinner：

```
$( ".selector" ).spinner({ culture: "fr" });
```

在初始化后，获取或设置 `culture` 选项：

```
// getter
var culture = $( ".selector" ).spinner( "option", "culture" );

// setter
$( ".selector" ).spinner( "option", "culture", "fr" );
```

disabledType: Boolean

Default: `false` 如果设置为 `true`，则禁用该 spinner（微调组件）。**Code examples:**

初始化带有指定 `disabled` 选项的 spinner：

```
$( ".selector" ).spinner({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).spinner( "option", "disabled" );

// setter
$( ".selector" ).spinner( "option", "disabled", true );
```

iconsType: Object

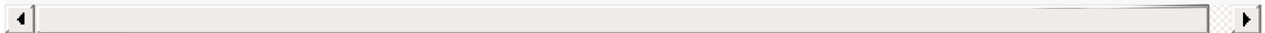
Default: `{ down: "ui-icon-triangle-1-s", up: "ui-icon-triangle-1-n" }` 标题要使用的图标，与 jQuery UI CSS 框架提供的图标（Icons） 匹配。设置为 false 则不显示图标。

- up (string, default: "ui-icon-triangle-1-n")
- down (string, default: "ui-icon-triangle-1-s")

Code examples:

初始化带有指定 `icons` 选项的 spinner：

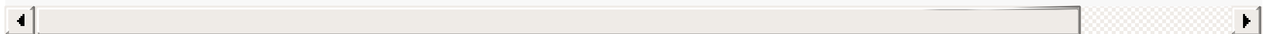
```
$( ".selector" ).spinner({ icons: { down: "custom-down-icon", up: "custom-up-icon" } });
```



在初始化后，获取或设置 `icons` 选项：

```
// getter
var icons = $( ".selector" ).spinner( "option", "icons" );

// setter
$( ".selector" ).spinner( "option", "icons", { down: "custom-down-icon", up: "custom-up-i
```



incrementalType: Boolean or Function(Integer count)

Default: `true` 当按住spinner（微调组件）按钮不放时，控制的步数。支持多个类型：

- **Boolean**:如果设置为 `true` ，当按住spinner（微调组件）按钮不放时，数值会根据 `step` 选项的值不断的增加或减少。 当设置为 `false` 时，所有步骤都是相等的，（由 `step` 选项所定义） （愚人码头注：点一下，数值会根据 `step` 选项的值增加或减少一步）。
- **Function**: 接收一个参数： 已发生的自旋数。 必须返回在当前发生的微调的步数。

Code examples:

初始化带有指定 `incremental` 选项的 spinner：

```
$( ".selector" ).spinner({ incremental: false });
```

在初始化后，获取或设置 `incremental` 选项：

```
// getter
var incremental = $( ".selector" ).spinner( "option", "incremental" );

// setter
$( ".selector" ).spinner( "option", "incremental", false );
```

maxType: Number or String

Default: `null` 允许的最大值。 如果元素的 `max` 属性存在，该选项未明确设置，那么该元素的 `max` 属性就被用作该选项的值。 如果为 `null`，表示没有上限。支持多个类型：

- **Number**: 最大值。
- **String**: 如果应用包含了Globalize， `max` 选项可以传递可以被 `numberFormat` 和 `culture` 选项解析的 字符串。 否则求助原生的 `parseFloat()` 将方法。

Code examples:

初始化带有指定 `max` 选项的 spinner：

```
$( ".selector" ).spinner({ max: 50 });
```

在初始化后，获取或设置 `max` 选项：

```
// getter
var max = $( ".selector" ).spinner( "option", "max" );

// setter
$( ".selector" ).spinner( "option", "max", 50 );
```

minType: Number or String

Default: `null` 允许的最小值。 如果元素的 `min` 属性存在，该选项未明确设置，那么该元素的 `min` 属性就被用作该选项的值。 如果为 `null`，表示没有下限。支持多个类型：

- **Number**: 最小值。
- **String**: 如果应用包含了Globalize， `max` 选项可以传递可以被 `numberFormat` 和 `culture` 选项解析的 字符串。 否则使用原生的 `parseFloat()` 方法解析。

Code examples:

初始化带有指定 `min` 选项的 spinner：

```
$( ".selector" ).spinner({ min: 0 });
```

在初始化后，获取或设置 `min` 选项：

```
// getter
var min = $( ".selector" ).spinner( "option", "min" );

// setter
$( ".selector" ).spinner( "option", "min", 0 );
```

numberFormatType: String

Default: `null` 通过 `Globalize` 格式化数字， 如果有效的话。 最常见的用于 `"n"` 用作十进制数 和 `"C"` 用作货币值。 也看到了 `culture` 选择。**Code examples:**

初始化带有指定 `numberFormat` 选项的 spinner：

```
$( ".selector" ).spinner({ numberFormat: "n" });
```

在初始化后，获取或设置 `numberFormat` 选项：

```
// getter
var numberFormat = $( ".selector" ).spinner( "option", "numberFormat" );

// setter
$( ".selector" ).spinner( "option", "numberFormat", "n" );
```

pageType: Number

Default: `10` 当通过 `pageUp` / `pageDown` 的方法进行分页时，采取的步数。**Code examples:**

初始化带有指定 `page` 选项的 spinner：

```
$( ".selector" ).spinner({ page: 5 });
```

在初始化后，获取或设置 `page` 选项：

```
// getter
var page = $( ".selector" ).spinner( "option", "page" );

// setter
$( ".selector" ).spinner( "option", "page", 5 );
```

stepType: Number or String

Default: `1` 通过按钮或 `stepUp()` / `stepDown()` 方法微调时，采取的步数。 如果元素的 `step` 属性存在，并且该选项未明确设置，那么元素的 `step` 属性值将作为该选项的值使用。支持多个类型：

- **Number**: 步数
- **String**: 如果应用包含了Globalize， `max` 选项可以传递可以被 `numberFormat` 和 `culture` 选项解析的 字符串。 否则使用原生的 `parseFloat()` 方法解析。

Code examples:

初始化带有指定 `step` 选项的 spinner：

```
$( ".selector" ).spinner({ step: 2 });
```

在初始化后，获取或设置 `step` 选项：

```
// getter
var step = $( ".selector" ).spinner( "option", "step" );

// setter
$( ".selector" ).spinner( "option", "step", 2 );
```

Methods

destroy()Returns: jQuery (plugin only)

完全移除 spinner功能。这会把元素返回到它的预初始化状态。

- 该方法不接受任何参数。

Code examples:

调用 destroy 方法：

```
$( ".selector" ).spinner( "destroy" );
```

disable()Returns: jQuery (plugin only)

禁用 spinner.

- 该方法不接受任何参数。

Code examples:

调用 disable 方法：

```
$( ".selector" ).spinner( "disable" );
```

enable()Returns: jQuery (plugin only)

启用 spinner.

- 该方法不接受任何参数。

Code examples:

调用 enable 方法：

```
$( ".selector" ).spinner( "enable" );
```

option(optionName)Returns: Object

获取当前与指定的 `optionName` 关联的值。

- **optionName**Type: String要获取值的选项的名称。

Code examples:

调用该方法：

```
var isDisabled = $( ".selector" ).spinner( "option", "disabled" );
```

option()Returns: PlainObject

获取一个包含键/值对的对象，键/值对表示当前 spinner 选项哈希。

- 该方法不接受任何参数。

Code examples:

调用该方法：

```
var options = $( ".selector" ).spinner( "option" );
```

option(optionName, value)Returns: jQuery (plugin only)

设置与指定的 `optionName` 关联的 spinner 选项的值。

- **optionName**Type: String要设置的选项的名称。
- **value**Type: Object要为选项设置的值。

Code examples:

调用该方法：

```
$( ".selector" ).spinner( "option", "disabled", true );
```

option(options)Returns: jQuery (plugin only)

为 spinner 设置一个或多个选项。

- **options**Type: Object要设置的 option-value 对。

Code examples:

调用该方法：

```
$( ".selector" ).spinner( "option", { disabled: true } );
```

pageDown([pages])Returns: jQuery (plugin only)

通过指定页数递减值， 页数由 `page` 选项定义。 如果没有参数， 单页递减。

如果返回值大于 `max` 选项定义的值，小于 `min` 选项定义的值，或返回的结果步数不匹配， 那么该值将被调整到最接近的有效值。

调用 `pageDown()` 将引起 `start` ， `spin` , 和 `stop` 事件被触发。

- **pages**Type: Number递减的页数，默认是1.

Code examples:

调用 pageDown 方法：

```
$( ".selector" ).spinner( "pageDown" );
```

pageUp([pages])Returns: jQuery (plugin only)

通过指定页数递增值， 页数由 `page` 选项定义。 如果没有参数， 单页递增。

如果返回值大于 `max` 选项定义的值，小于 `min` 选项定义的值，或返回的结果步数不匹配， 那么该值将被调整到最接近的有效值。

调用 `pageUp()` 将引起 `start` ， `spin` , 和 `stop` 事件被触发。

- **pages**Type: Number递增的页数，默认是1.

Code examples:

调用 pageUp 方法：

```
$( ".selector" ).spinner( "pageUp", 10 );
```

stepDown([steps])Returns: jQuery (plugin only)

通过指定步数递减值， 步数由 `step` 选项定义。 如果没有参数， 单步递减。

如果返回值大于 `max` 选项定义的值，小于 `min` 选项定义的值，或返回的结果步数不匹配， 那么该值将被调整到最接近的有效值。

调用 `stepDown()` 将引起 `start` , `spin` , 和 `stop` 事件被触发。

- **steps**Type: Number递减的步数， 默认是1.

Code examples:

调用 stepDown 方法：

```
$( ".selector" ).spinner( "stepDown" );
```

stepUp([steps])Returns: jQuery (plugin only)

通过指定步数递增值， 步数由 `step` 选项定义。 如果没有参数， 单步递增。

如果返回值大于 `max` 选项定义的值，小于 `min` 选项定义的值，或返回的结果步数不匹配， 那么该值将被调整到最接近的有效值。

调用 `pageUp()` 将引起 `start` , `spin` , 和 `stop` 事件被触发。

- **steps**Type: Number递增的步数， 默认是1.

Code examples:

调用 stepUp 方法：

```
$( ".selector" ).spinner( "stepUp", 5 );
```

value()Returns: Number

获取当前数值，这个值是基于 `numberFormat` 和 `culture` 选项解析的。

- 该方法不接受任何参数。

Code examples:

调用该方法：

```
var value = $( ".selector" ).spinner( "value" );
```

value(value)Returns: jQuery (plugin only)

设置当前值

- **value**Type: Number or String用来设置的值。 如果传递的是一个字符串，那么 这个值是基于 `numberFormat` 和 `culture` 选项解析的。

Code examples:

调用该方法：

```
$( ".selector" ).spinner( "value", 50 );
```

widget()Returns: jQuery

返回包含生成组件包裹元素 的一个 `jQuery` 对象。

- 该方法不接受任何参数。

Code examples:

调用 widget 方法：

```
var widget = $( ".selector" ).spinner( "widget" );
```

扩展点（Extension Points）

spinner（微调组件）是widget factory构建的，并且可以扩展。 当扩展部件时， 你必须覆盖或添加新的行为到现有的方法。 下列方法被提供作为扩展点 用相同的API稳定性如上所列的plugin methods。 有关小部件扩展的更多信息， 请参阅使用Widget Factory 扩展小部件。

_buttonHtml()Returns: String

这个方法返回的HTML用于spinner（微调组件）的递增和递减按钮。 每个按钮都必须给定一个 `ui-spinner-button` 的类名 用于相关联的事件工作。

- 该方法不接受任何参数。

Code examples:

使用 <button> 元素 作为递增和递减按钮。

```
_buttonHtml: function() {
  return "" +
    "<button class='ui-spinner-button ui-spinner-up'>" +
      "<span class='ui-icon " + this.options.icons.up + "'>&#9650;</span>" +
    "</button>" +
    "<button class='ui-spinner-button ui-spinner-down'>" +
      "<span class='ui-icon " + this.options.icons.down + "'>&#9660;</span>" +
    "</button>";
}
```

_uiSpinnerHtml()Returns: String

这个方法返回的HTML用于包裹 spinner（微调组件） <input> 元素。

- 该方法不接受任何参数。

Code examples:

用没有圆角的 <div> 包裹 spinner（微调组件）。

```
_uiSpinnerHtml: function() {
  return "<div class='ui-spinner ui-widget ui-widget-content'></div>";
}
```

Events

change(event, ui)Type: spinchange

当spinner微调器的值改变并且输入元素（input）失去焦点时，该事件触发。

- **event**Type: Event
- **ui**Type: Object

注意： *ui* 对象是空的，这里包含它是为了与其他事件保持一致性。

Code examples:

初始化带有指定 change 回调的 spinner（微调器）：

```
$( ".selector" ).spinner({
  change: function( event, ui ) {}
});
```

绑定一个事件监听器到 spinchange 事件：

```
$( ".selector" ).on( "spinchange", function( event, ui ) {} );
```

create(event, ui)Type: `spincreate`

当spinner微调器创建的时候，该时间触发。

- **event**Type: Event
- **ui**Type: Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

Code examples:

初始化带有指定 create 回调的 spinner（微调器）：

```
$( ".selector" ).spinner({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 spincreate 事件：

```
$( ".selector" ).on( "spincreate", function( event, ui ) {} );
```

spin(event, ui)Type: `spin`

在递增/递减的时候，该事件触发（用 当前值和 `ui.value` 比较来 确定的微调的方向）。

可以取消，以防止被更新值。

- **event**Type: Event
- **ui**Type: Object
    - **value**Type: Number要设置的新值，除非该事件被取消。/div>

Code examples:

初始化带有指定 spin 回调的 spinner（微调器）：

```
$( ".selector" ).spinner({
  spin: function( event, ui ) {}
});
```

绑定一个事件监听器到 spin 事件：

```
$( ".selector" ).on( "spin", function( event, ui ) {} );
```

start(event, ui)Type: `spinstart`

微调开始之前，触发该事件。可以取消，以防止微调。

- **event**Type: Event
- **ui**Type: Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

Code examples:

初始化带有指定 start 回调的 spinner（微调器）：

```
$( ".selector" ).spinner({
  start: function( event, ui ) {}
});
```

绑定一个事件监听器到 spinstart 事件：

```
$( ".selector" ).on( "spinstart", function( event, ui ) {} );
```

stop(event, ui)Type: `spinstop`

微调结束后，触发该事件。

- **event**Type: Event
- **ui**Type: Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

Code examples:

初始化带有指定 stop 回调的 spinner（微调器）：

```
$( ".selector" ).spinner({
  stop: function( event, ui ) {}
});
```

绑定一个事件监听器到 spinstop 事件：

```
$( ".selector" ).on( "spinstop", function( event, ui ) {} );
```

Example:

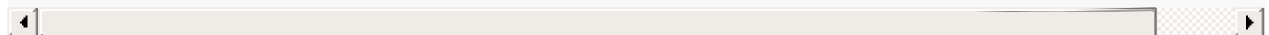
普通的数字微调器。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>spinner demo</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-ui.cs
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<input id="spinner">

<script>
$( "#spinner" ).spinner();
</script>

</body>
</html>
```



Tabs Widget

Categories: Widgets

version added: 1.0

Description: 一种多panel（面板）的单内容区，每个panel（面板）与列表中的标题相关。

QuickNavExamples

Options

- active
- collapsible
- disabled
- event
- heightStyle
- hide
- show

Methods

- destroy
- disable
- enable
- load
- option
- refresh
- widget

Events

- activate
- beforeActivate
- beforeLoad
- create
- load

选项卡（Tabs）通常用于把内容分成多个部分，以便节省空间，就像折叠面板（accordion）一样。

选项卡（Tabs）有一组必须使用的特定标记，以便选项卡能正常工作：

- 选项卡（Tabs）必须在一个有序的（ ）或无序的（ ）列表中
- 每个选项卡的 "title" 必须在一个列表项（ ）的内部，且必须用一个带有 `href` 属性的锚（ <a> ）包裹。
- 每个选项卡panel（面板）可以是任意有效的元素，但是它必须带有一个 id，该 id 与相关选项卡的锚中的哈希相对应。

每个选项卡panel（面板）的内容可以在页面中定义好，或者可以通过 Ajax 加载。这两种方式都是基于与选项卡相关的锚的 `href` 上自动处理的。默认情况下，选项卡在点击时激活，但是通过 `event` 选项可以改变或覆盖事件。

下面是一些样品标记：

```
<div id="tabs">
  <ul>
    <li><a href="#fragment-1">One</a></li>
    <li><a href="#fragment-2">Two</a></li>
    <li><a href="#fragment-3">Three</a></li>
  </ul>
  <div id="fragment-1">
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euism
  </div>
  <div id="fragment-2">
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euism
  </div>
  <div id="fragment-3">
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euism
  </div>
</div>
```

键盘交互（Keyboard interaction）

当焦点在选项卡上时，下面的键盘命令可用：

- UP/LEFT：移动焦点到上一个选项卡。如果在第一个选项卡上，则移动焦点到最后一个选项卡。在一个短暂的延迟后激活获得焦点的选项卡。
- DOWN/RIGHT：移动焦点到下一个选项卡。如果在最后一个选项卡上，则移动焦点到第一个选项卡。在一个短暂的延迟后激活获得焦点的选项卡。
- HOME：移动焦点到第一个选项卡。在一个短暂的延迟后激活获得焦点的选项卡。
- END：移动焦点到最后一个选项卡。在一个短暂的延迟后激活获得焦点的选项卡。
- SPACE：激活与获得焦点的选项卡相关的panel（面板）。
- ENTER：激活或切换与获得焦点的选项卡相关的panel（面板）。
- ALT+PAGE UP：移动焦点到上一个选项卡，并立即激活。
- ALT+PAGE DOWN：移动焦点到下一个选项卡，并立即激活。

当焦点在panel（面板）上时，下面的键盘命令可用：

- CTRL+UP：移动焦点到相关的选项卡。
- ALT+PAGE UP：移动焦点到上一个选项卡，并立即激活。
- ALT+PAGE DOWN：移动焦点到下一个选项卡，并立即激活。

主题（Theming）

选项卡部件（Tabs Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用选项卡指定的样式，则可以使用下面的 CSS class 名称：

- `ui-tabs`：选项卡的外层容器。当设置了 `collapsible` 选项时，该元素会另外带有一个 `ui-tabs-collapsible` class。
    - `ui-tabs-nav`：选项卡列表。
        - 导航中激活的列表项会带有一个 `ui-tabs-active` class。内容通过 Ajax 调用加载的列表项会带有一个 `ui-tabs-loading` class。
            - `ui-tabs-anchor`：用于切换panel（面板）的锚。
    - `ui-tabs-panel`：与选项卡相关的panel（面板）。只有与其对应的选项卡激活时才可见。

依赖（Dependencies）

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 特效核心（Effects Core）（可选的；当与 `show` 和 `hide` 选项一起使用时）

Additional Notes:

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

Options

activeType: Boolean or Integer

Default: `0` 当前打开哪一个panel（面板）。支持多个类型：

- **Boolean**:设置 `active` 为 `false` 将折叠所有的panel（面板）。这要求 `collapsible` 选项必须为 `true` 。
- **Integer**: 激活打开的panel（面板）索引，以零为基础。负值则表示从最后一个panel（面板）后退选择panel（面板）。

Code examples:

初始化带有指定 `active` 选项的 Tab（选项卡）：

```
$( ".selector" ).tabs({ active: 1 });
```

在初始化后，获取或设置 `active` 选项：

```
// getter
var active = $( ".selector" ).tabs( "option", "active" );

// setter
$( ".selector" ).tabs( "option", "active", 1 );
```

collapsibleType: Boolean

Default: `false` 当设置为 `true` 时，激活的panel（面板）可以被关闭。**Code examples:**

初始化带有指定 `collapsible` 选项的 Tab（选项卡）：

```
$( ".selector" ).tabs({ collapsible: true });
```

在初始化后，获取或设置 `collapsible` 选项：

```
// getter
var collapsible = $( ".selector" ).tabs( "option", "collapsible" );

// setter
$( ".selector" ).tabs( "option", "collapsible", true );
```

disabledType: Boolean or Array

Default: `false` 哪些标签被禁用。支持多个类型：

- **Boolean**: 启用或禁用所有选项卡。
- **Array**: 一个数组。包含从零开始计数的应该禁用选项卡的索引，例如，`[ 0, 2 ]` 将禁用第一和第三个选项卡。

Code examples:

初始化带有指定 `disabled` 选项的 Tab（选项卡）：

```
$( ".selector" ).tabs({ disabled: [ 0, 2 ] });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).tabs( "option", "disabled" );

// setter
$( ".selector" ).tabs( "option", "disabled", [ 0, 2 ] );
```

eventType: String

Default: `"click"` 激活选项卡的事件类型。 要悬停（hover）激活选项卡，用 `"mouseover"` 。**Code examples:**

初始化带有指定 `event` 选项的 Tab（选项卡）：

```
$( ".selector" ).tabs({ event: "mouseover" });
```

在初始化后，获取或设置 `event` 选项：

```
// getter
var event = $( ".selector" ).tabs( "option", "event" );

// setter
$( ".selector" ).tabs( "option", "event", "mouseover" );
```

heightStyleType: String

Default: `"content"` 控制Tab（选项卡）部件和每个panel（面板）的高度。 可能的值：

- `"auto"` : 所有的panel（面板）将会被设置为最高的panel（面板）的高度。
- `"fill"` : 基于 tabs 的父元素的高度，扩展到可用的高度。
- `"content"` : 每个panel（面板）的高度取决于它的内容。

Code examples:

初始化带有指定 `heightStyle` 选项的 Tab（选项卡）：

```
$( ".selector" ).tabs({ heightStyle: "fill" });
```

在初始化后，获取或设置 `heightStyle` 选项：

```
// getter
var heightStyle = $( ".selector" ).tabs( "option", "heightStyle" );

// setter
$( ".selector" ).tabs( "option", "heightStyle", "fill" );
```

hideType: Boolean or Number or String or Object

Default: `null` panel（面板）隐藏时的动画效果。支持多个类型：

- **Boolean**: 当设置为 `false`， 将不使用动画效果，该panel（面板）会立即被隐藏。 如果设置为 `true`， 该panel（面板）将会以默认的持续时间和默认的效果淡出。
- **Number**: 该panel（面板）将以指定的时间和默认的效果淡出。
- **String**: 该panel（面板）将使用指定的效果被隐藏。 该值可以是一个jQuery内置的动画方法的名称， 如 `"slideUp"`， 或一个 jQuery UI 效果的名称， 如 `"fold"` 。 在这两种情况下，将使用默认持续时间和默认的动画效果。
- **Object**: 如果该值是一个对象， 那么 `effect`, `delay`, `duration`, 和 `easing` 可能要提供。 如果 `effect` 属性包含一个jQuery方法的名称， 那么该方法将被使用; 否则它被假定为是一个jQuery UI的效果的名称。 当使用jQuery UI 支持额外设置 的效果， 你可以在对象中包含那些设置 并且它们将被传递到的效果。如果 `duration` 持续时间或 `easing` 属性被省略， 那么默认值将被使用。 如果 `effect` 被省略， 那么 `"fadeOut"` 将被使用。如果 `delay` 被省略， 那么将不使用延迟。

Code examples:

初始化带有指定 `hide` 选项的 Tab（选项卡）：

```
$( ".selector" ).tabs({ hide: { effect: "explode", duration: 1000 } });
```

在初始化后，获取或设置 `hide` 选项：

```
// getter
var hide = $( ".selector" ).tabs( "option", "hide" );

// setter
$( ".selector" ).tabs( "option", "hide", { effect: "explode", duration: 1000 } );
```

showType: Boolean or Number or String or Object

Default: `null` panel（面板）显示时的动画效果。支持多个类型：

- **Boolean**: 当设置为 `false`， 将不使用动画效果，该panel（面板）会立即被隐藏。 如果设置为 `true`， 该panel（面板）将会以默认的持续时间和默认的效果淡出。
- **Number**: 该panel（面板）将以指定的时间和默认的效果淡出。
- **String**: 该panel（面板）将使用指定的效果被隐藏。 该值可以是一个jQuery内置的动画方法的名称， 如 `"slideUp"`， 或一个 jQuery UI 效果的名称， 如 `"fold"` 。 在这两种情况下，将使用默认持续时间和默认的动画效果。
- **Object**: 如果该值是一个对象， 那么 `effect`, `delay`, `duration`, 和 `easing` 可能要提供。 如果 `effect` 属性包含一个jQuery方法的名称， 那么该方法将被使用; 否则它被假定为是一个jQuery UI的效果的名称。 当使用jQuery UI 支持额外设置 的效果， 你可以在对象中包含那些设置 并且它们将被传递到的效果。如果 `duration` 持续时间或 `easing` 属性被省略， 那么默认值将被使用。 如果 `effect` 被省略， 那么 `"fadeOut"` 将被使用。如

果 `delay` 被省略，那么将不使用延迟。

Code examples:

初始化带有指定 `show` 选项的 Tab（选项卡）：

```
$( ".selector" ).tabs({ show: { effect: "blind", duration: 800 } });
```

在初始化后，获取或设置 `show` 选项：

```
// getter
var show = $( ".selector" ).tabs( "option", "show" );

// setter
$( ".selector" ).tabs( "option", "show", { effect: "blind", duration: 800 } );
```

Methods

destroy()Returns: jQuery (plugin only)

完全移除 Tab（选项卡）功能. 这将使元素返回到之前的初始化状态。

- 该方法不接受任何参数。

Code examples:

调用 destroy 方法：

```
$( ".selector" ).tabs( "destroy" );
```

disable()Returns: jQuery (plugin only)

禁用全部的 tabs（选项卡）。

- 该方法不接受任何参数。

Code examples:

调用该方法：

```
$( ".selector" ).tabs( "disable" );
```

disable(index)Returns: jQuery (plugin only)

禁用某个 tabs（选项卡）。选定的选项卡不能被禁用。 要一次禁用多个选项卡， 设置 `disabled` 选项： `$( "#tabs" ).tabs( "option", "disabled", [ 1, 2, 3 ] )`.

- **index**Type: Number or String哪个 tabs（选项卡）被禁用。

Code examples:

调用该方法：

```
$( ".selector" ).tabs( "disable", 1 );
```

enable()Returns: jQuery (plugin only)

启用全部的 tabs（选项卡）。

- 该方法不接受任何参数。

Code examples:

调用该方法：

```
$( ".selector" ).tabs( "enable" );
```

enable(index)Returns: jQuery (plugin only)

启用某个 tabs（选项卡）。 要一次要启多个选项卡，可以重置disabled属性，如： `$( "#example" ).tabs( "option", "disabled", [] );`.

- **index**Type: Number or StringWhich tab to enable.

Code examples:

调用该方法：

```
$( ".selector" ).tabs( "enable", 1 );
```

load(index)Returns: jQuery (plugin only)

加载远程选项卡的面板内容。

- **index**Type: Number or String哪个tabs需要加载

Code examples:

调用 load 方法：

```
$( ".selector" ).tabs( "load", 1 );
```

option(optionName)Returns: Object

获取当前与指定的 `optionName` 关联的值。

- **optionName**Type: String要获取值的选项的名称。

Code examples:

调用该方法：

```
var isDisabled = $( ".selector" ).tabs( "option", "disabled" );
```

option()Returns: PlainObject

获取一个包含键/值对的对象，键/值对表示当前 tabs 选项哈希。

- 该方法不接受任何参数。

Code examples:

调用该方法：

```
var options = $( ".selector" ).tabs( "option" );
```

option(optionName, value)Returns: jQuery (plugin only)

设置与指定的 `optionName` 关联的 tabs 选项的值。

- **optionName**Type: String要设置的选项的名称。
- **value**Type: Object要为选项设置的值。

Code examples:

调用该方法：

```
$( ".selector" ).tabs( "option", "disabled", true );
```

option(options)Returns: jQuery (plugin only)

为 tabs（选项卡） 设置一个或多个选项。

- **options**Type: Object要设置的 option-value 对。

Code examples:

调用该方法：

```
$( ".selector" ).tabs( "option", { disabled: true } );
```

refresh()Returns: jQuery (plugin only)

处理任何在 DOM 中直接添加或移除的标题和面板，并重新计算 tabs（选项卡） 的高度。结果取决于内容和 `heightStyle` 选项。

- 该方法不接受任何参数。

Code examples:

调用 refresh 方法：

```
$( ".selector" ).tabs( "refresh" );
```

widget()Returns: jQuery

返回一个包含 tabs（选项卡） 的 `jQuery` 对象。

- 该方法不接受任何参数。

Code examples:

调用 widget 方法：

```
var widget = $( ".selector" ).tabs( "widget" );
```

Events

activate(event, ui)Type: `tabsactivate`

面板被激活后触发（在动画完成之后）。如果 tabs（选项卡） 之前是关闭的，则 `ui.oldTab` 和 `ui.oldPanel` 将是空的 jQuery 对象。如果 tabs（选项卡） 正在折叠，则 `ui.newTab` 和 `ui.newPanel` 将是空的 jQuery 对象。

注意： 由于 `activate` 事件只有在面板激活时才能触发，当创建 tabs（选项卡） 部件时，最初的面板不会触发该事件。如果您需要一个用于部件创建的钩，请使用 `create` 事件。

- **event**Type: Event
- **ui**Type: Object

  - **newTab**Type: jQuery刚被激活的选项卡。
  - **oldTab**Type: jQuery刚被取消激活的选项卡。
  - **newPanel**Type: jQuery刚被激活的面板。
  - **oldPanel**Type: jQuery刚被取消激活的面板。

Code examples:

初始化带有指定 activate 回调的 Tab（选项卡面板）：

```
$( ".selector" ).tabs({
  activate: function( event, ui ) {}
});
```

绑定一个事件监听器到 tabsactivate 事件：

```
$( ".selector" ).on( "tabsactivate", function( event, ui ) {} );
```

beforeActivate(event, ui)Type: `tabsbeforeactivate`

tabs（选项卡）被激活前直接触发。可以取消以防止tabs（选项卡）被激活。如果 tabs（选项卡） 当前是关闭的，则 `ui.oldTab` 和 `ui.oldPanel` 将是空的 jQuery 对象。如果 accordion 正在折叠，则 `ui.newTab` 和 `ui.newPanel` 将是空的 jQuery 对象。

- **event**Type: Event
- **ui**Type: Object
  - **newTab**Type: jQuery刚被激活的选项卡。
  - **oldTab**Type: jQuery刚被取消激活的选项卡。
  - **newPanel**Type: jQuery刚被激活的面板。
  - **oldPanel**Type: jQuery刚被取消激活的面板。

Code examples:

初始化带有指定 beforeActivate 回调的 Tab（选项卡面板）：

```
$( ".selector" ).tabs({
  beforeActivate: function( event, ui ) {}
});
```

绑定一个事件监听器到 tabsbeforeactivate 事件：

```
$( ".selector" ).on( "tabsbeforeactivate", function( event, ui ) {} );
```

beforeLoad(event, ui)Type: `tabsbeforeload`

在一个远程选项卡被加载之前， `beforeActivate` 事件之后，触发该事件。可以取消，以防止tabs（选项卡）面板加载内容;虽然面板仍然会被激活。 该事件被触发Ajax请求发送之前，这样可以使用 `ui.jqXHR` 和 `ui.ajaxSettings` 修改。

注意： 虽然 *`ui.ajaxSettings`* 被提供，并且可以被修改， 其中的一些设置已经通过*jQuery*处理。 例如，*prefilters*已经被应用， *`data`* 已被处理，还有 *`type`* 已经确定。 ， 因为 *`beforeSend`* 作为 *`jQuery.ajax()`* 的回调， *`beforeLoad`* 事件同时是发生的，因此具有相同的限制。

- **event**Type: Event
- **ui**Type: Object
  - **tab**Type: jQuery正在加载的选项卡。
  - **panel**Type: jQuery将Ajax响应填充的面板。
  - **jqXHR**Type: jqXHR被请求内容的 `jqXHR` 对象。
  - **ajaxSettings**Type: Object用于由 `jQuery.ajax` 请求内容的设置。

Code examples:

初始化带有指定 beforeLoad 回调的 Tab（选项卡面板） ：

```
$( ".selector" ).tabs({
  beforeLoad: function( event, ui ) {}
});
```

绑定一个事件监听器到 tabsbeforeload 事件：

```
$( ".selector" ).on( "tabsbeforeload", function( event, ui ) {} );
```

create(event, ui)Type: `tabscreate`

当创建 tabs（选项卡) 时触发。如果 tabs（选项卡) 是关闭的， `ui.tab` 和 `ui.panel` 将是空的 jQuery 对象。

- **event**Type: Event
- **ui**Type: Object
  - **tab**Type: jQuery激活的选项卡
  - **panel**Type: jQuery激活的面板。

Code examples:

初始化带有指定 create 回调的 Tab（选项卡面板） ：

```
$( ".selector" ).tabs({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 tabscreate 事件：

```
$( ".selector" ).on( "tabscreate", function( event, ui ) {} );
```

load(event, ui)Type: `tabsload`

远程选项卡加载后触发该事件。

- **event**Type: Event
- **ui**Type: Object
  - **tab**Type: jQuery刚加载的选项卡。
  - **panel**Type: jQuery刚刚填充的Ajax响应的面板。

Code examples:

初始化带有指定 load 回调的 Tab（选项卡面板）：

```
$( ".selector" ).tabs({
  load: function( event, ui ) {}
});
```

绑定一个事件监听器到 tabsload 事件：

```
$( ".selector" ).on( "tabsload", function( event, ui ) {} );
```

Example:

一个简单的 **jQuery UI** 选项卡（**Tabs**）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>tabs demo</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-ui.cs
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="tabs">
  <ul>
    <li><a href="#fragment-1"><span>One</span></a></li>
    <li><a href="#fragment-2"><span>Two</span></a></li>
    <li><a href="#fragment-3"><span>Three</span></a></li>
  </ul>
  <div id="fragment-1">
    <p>First tab is active by default:</p>
    <pre><code>$( "#tabs" ).tabs(); </code></pre>
  </div>
  <div id="fragment-2">
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euism
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euism
  </div>
  <div id="fragment-3">
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euism
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euism
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euism
  </div>
</div>

<script>
$( "#tabs" ).tabs();
</script>

</body>
</html>
```

Tooltip Widget

Categories: Widgets

version added: 1.9

Description: 可自定义的、可主题化的工具提示框，替代原生的工具提示框。

QuickNavExamples

Options

- content
- disabled
- hide
- items
- position
- show
- tooltipClass
- track

Methods

- close
- destroy
- disable
- enable
- open
- option
- widget

Events

- close
- create
- open

工具提示框（Tooltip）取代了原生的工具提示框，让它们可主题化，也允许进行各种自定义：

- 显示不仅仅是标题的其他内容，就如内联的脚注或通过 Ajax 检索的额外内容。
- 自定义定位，例如，在元素上居中工具提示框。
- 添加额外的样式来定制警告或错误区域的外观。

默认使用一个渐变的动画来显示和隐藏工具提示框，这种外观与简单的切换可见度相比更具灵性。这可以通过 `show` 和 `hide` 选项进行定制。

- `items` 和 `content` 选项需要保持同步。如果您改变了其中一个，您需要同时改变另一个。

在一般情况下，禁用的元素不会触发任何 DOM 事件。因此，适当地控制禁用元素的工具提示框是不可能的，因为我们需要监听事件来决定何时显示和隐藏工具提示框。这就导致 jQuery UI 不能保证对附加到禁用元素上的工具提示框任何层次上的支持。这意味着如果您需要在禁用元素上进行提示，您可能需要使用一个原生的提示框和 jQuery UI 工具提示框的混合物。

主题（Theming）

工具提示框部件（Tooltip Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用工具提示框指定的样式，则可以使用下面的 CSS class 名称：

- `ui-tooltip`：工具提示框的外层容器。
  - `ui-tooltip-content`：工具提示框的内容。

依赖（Dependencies）

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 定位（Position）
- 特效核心（Effects Core）（可选的；当与 `show` 和 `hide` 选项一起使用时）

其他注意事项（Additional Notes）：

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

Options

contentType: Function() or String

Default: `function returning the title attribute`

tooltip(工具提示框)的内容。

当改变这个选项时，你可能还需要更改 `items` 选项。

支持多个类型：

- **Function**: 一个回调可以是直接返回的内容， 或通过在内容，调用第一个参数，例如，对于Ajax内容。
- **String**: 一个HTML字符串作为tooltip(工具提示框)的内容。

Code examples:

初始化带有指定 `content` 选项的 tooltip(工具提示框)：

```
$( ".selector" ).tooltip({ content: "Awesome title!" });
```

在初始化后，获取或设置 `content` 选项：

```
// getter
var content = $( ".selector" ).tooltip( "option", "content" );

// setter
$( ".selector" ).tooltip( "option", "content", "Awesome title!" );
```

disabledType: Boolean

Default: `false` 如果设置为 `true`， 则禁用该 tooltip（工具提示框）。**Code examples:**

初始化带有指定 `disabled` 选项的 tooltip(工具提示框)：

```
$( ".selector" ).tooltip({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).tooltip( "option", "disabled" );

// setter
$( ".selector" ).tooltip( "option", "disabled", true );
```

hideType: Boolean or Number or String or Object

Default: `true` tooltip(工具提示框)关闭（隐藏）时的动画效果。支持多个类型：

- **Boolean**: 当设置为 `false`， 将不使用动画效果，该 tooltip(工具提示框) 会立即被隐藏。如果设置为 `true`， 该 tooltip(工具提示框) 将会以默认的持续时间和默认的效果淡出。
- **Number**: 该 tooltip(工具提示框) 将以指定的时间和默认的效果淡出。
- **String**: 该 tooltip(工具提示框) 将使用指定的效果被隐藏。 该值可以是一个jQuery内置的动画方法的名称， 如 `"slideUp"`， 或一个 jQuery UI 效果的名称， 如 `"fold"`。 在这两

种情况下，将使用默认持续时间和默认的动画效果。
- **Object**: 如果该值是一个对象， 那么 `effect` , `delay` , `duration` , 和 `easing` 可能要提供。 如果 `effect` 属性包含一个jQuery方法的名称， 那么该方法将被使用; 否则它被假定为是一个jQuery UI的效果的名称。 当使用jQuery UI 支持额外设置 的效果， 你可以在对象中包含那些设置 并且它们将被传递到的效果。如果 `duration` 持续时间或 `easing` 属性被省略， 那么默认值将被使用。 如果 `effect` 被省略， 那么 `"fadeOut"` 将被使用。如果 `delay` 被省略， 那么将不使用延迟。

Code examples:

初始化带有指定 `hide` 选项的 tooltip(工具提示框)：

```
$( ".selector" ).tooltip({ hide: { effect: "explode", duration: 1000 } });
```

在初始化后，获取或设置 `hide` 选项：

```
// getter
var hide = $( ".selector" ).tooltip( "option", "hide" );

// setter
$( ".selector" ).tooltip( "option", "hide", { effect: "explode", duration: 1000 } );
```

itemsType: Selector

Default: `[title]`

一个选择器表示哪些项目应该显示tooltip(工具提示框)。 如果您使用其他的东西自定义，那么title属性将作为tooltip(工具提示框)的内容， 或者你需要一个不同的选择来事件委托。

当改变这个选项时，你可能还需要改变的 `content` 选项。

Code examples:

初始化带有指定 `items` 选项的 tooltip(工具提示框)：

```
$( ".selector" ).tooltip({ items: "img[alt]" });
```

在初始化后，获取或设置 `items` 选项：

```
// getter
var items = $( ".selector" ).tooltip( "option", "items" );

// setter
$( ".selector" ).tooltip( "option", "items", "img[alt]" );
```

positionType: Object

Default: `{ my: "left top+15", at: "left bottom", collision: "flipfit" }`

确定 tooltip(工具提示框) 相对于 相关目标元素的位置。 `of` 选项默认为目标元素， 但您可以指定其他元素来定位。 有关各个选项的更多细节， 你可以参考jQuery UI Position实用工具。

Code examples:

初始化带有指定 `position` 选项的 tooltip(工具提示框)：

```
$( ".selector" ).tooltip({ position: { my: "left+15 center", at: "right center" } });
```

在初始化后，获取或设置 `position` 选项：

```
// getter
var position = $( ".selector" ).tooltip( "option", "position" );

// setter
$( ".selector" ).tooltip( "option", "position", { my: "left+15 center", at: "right center
```

showType: Boolean or Number or String or Object

Default: `true` tooltip(工具提示框) 打开（显示）时的动画效果。支持多个类型：

- **Boolean**: 当设置为 `false`， 将不使用动画效果，该 tooltip(工具提示框) 会立即被隐藏。如果设置为 `true`， 该 tooltip(工具提示框) 将会以默认的持续时间和默认的效果淡出。
- **Number**: 该 tooltip(工具提示框) 将以指定的时间和默认的效果淡出。
- **String**: 该 tooltip(工具提示框) 将使用指定的效果被隐藏。 该值可以是一个jQuery内置的动画方法的名称， 如 `"slideUp"`， 或一个 jQuery UI 效果的名称， 如 `"fold"`。 在这两种情况下，将使用默认持续时间和默认的动画效果。
- **Object**: 如果该值是一个对象， 那么 `effect`, `delay`, `duration`, 和 `easing` 可能要提供。 如果 `effect` 属性包含一个jQuery方法的名称， 那么该方法将被使用; 否则它被假定为是一个jQuery UI的效果的名称。 当使用jQuery UI 支持额外设置 的效果， 你可以在对象中包含那些设置 并且它们将被传递到的效果。如果 `duration` 持续时间或 `easing` 属性被省略， 那么默认值将被使用。 如果 `effect` 被省略， 那么 `"fadeOut"` 将被使用。如果 `delay` 被省略， 那么将不使用延迟。

Code examples:

初始化带有指定 `show` 选项的 tooltip(工具提示框)：

```
$( ".selector" ).tooltip({ show: { effect: "blind", duration: 800 } });
```

在初始化后，获取或设置 `show` 选项：

```
// getter
var show = $( ".selector" ).tooltip( "option", "show" );

// setter
$( ".selector" ).tooltip( "option", "show", { effect: "blind", duration: 800 } );
```

tooltipClassType: String

Default: `null` 一个CSS样式类名 添加到tooltip(工具提示框)小部件， 可用于显示各种提示类型， 像警告或错误。

这可能会被classes option取代。

Code examples:

初始化带有指定 `tooltipClass` 选项的 tooltip(工具提示框)：

```
$( ".selector" ).tooltip({ tooltipClass: "custom-tooltip-styling" });
```

在初始化后，获取或设置 `tooltipClass` 选项：

```
// getter
var tooltipClass = $( ".selector" ).tooltip( "option", "tooltipClass" );

// setter
$( ".selector" ).tooltip( "option", "tooltipClass", "custom-tooltip-styling" );
```

trackType: Boolean

Default: `false` tooltip(工具提示框)是否应该跟踪（跟随）鼠标。**Code examples:**

初始化带有指定 `track` 选项的 tooltip(工具提示框)：

```
$( ".selector" ).tooltip({ track: true });
```

在初始化后，获取或设置 `track` 选项：

```
// getter
var track = $( ".selector" ).tooltip( "option", "track" );

// setter
$( ".selector" ).tooltip( "option", "track", true );
```

Methods

close()Returns: jQuery (plugin only)

关闭 tooltip(工具提示框)。这仅供非委派的 tooltip(工具提示框) 调用。

- 该方法不接受任何参数。

Code examples:

调用 close 方法：

```
$( ".selector" ).tooltip( "close" );
```

destroy()Returns: jQuery (plugin only)

完全移除 tooltip(工具提示框) 功能. 这将使元素返回到之前的初始化状态.

- 该方法不接受任何参数。

Code examples:

调用 destroy 方法：

```
$( ".selector" ).tooltip( "destroy" );
```

disable()Returns: jQuery (plugin only)

禁用 tooltip(工具提示框)。

- 该方法不接受任何参数。

Code examples:

调用 disable 方法：

```
$( ".selector" ).tooltip( "disable" );
```

enable()Returns: jQuery (plugin only)

启用 tooltip(工具提示框)。

- 该方法不接受任何参数。

Code examples:

调用 enable 方法：

```
$( ".selector" ).tooltip( "enable" );
```

open()Returns: jQuery (plugin only)

以编程方式打开一个 tooltip(工具提示框)。这仅供非委派的 tooltip(工具提示框) 调用。

- 该方法不接受任何参数。

Code examples:

调用 open 方法：

```
$( ".selector" ).tooltip( "open" );
```

option(optionName)Returns: Object

获取当前与指定的 `optionName` 关联的值。

- **optionName**Type: String要获取值的选项的名称。

Code examples:

调用该方法：

```
var isDisabled = $( ".selector" ).tooltip( "option", "disabled" );
```

option()Returns: PlainObject

获取一个包含键/值对的对象，键/值对表示当前 tooltip(工具提示框) 选项哈希。

- 该方法不接受任何参数。

Code examples:

调用该方法：

```
var options = $( ".selector" ).tooltip( "option" );
```

option(optionName, value)Returns: jQuery (plugin only)

设置与指定的 `optionName` 关联的 tooltip(工具提示框) 选项的值。

- **optionName**Type: String要设置的选项的名称。
- **value**Type: Object要为选项设置的值。

Code examples:

调用该方法：

```
$( ".selector" ).tooltip( "option", "disabled", true );
```

option(options)Returns: jQuery (plugin only)

为 tooltip(工具提示框) 设置一个或多个选项。

- **options**Type: Object要设置的 option-value 对。

Code examples:

调用该方法：

```
$( ".selector" ).tooltip( "option", { disabled: true } );
```

widget()Returns: jQuery

返回一个包含 生成包裹元素 的 `jQuery` 对象。

- 该方法不接受任何参数。

Code examples:

调用 widget 方法：

```
var widget = $( ".selector" ).tooltip( "widget" );
```

Events

close(event, ui)Type: `tooltipclose`

当 tooltip(工具提示框) 关闭时触发，触发事件为 `focusout` 或 `mouseleave` 。

- **event**Type: Event
- **ui**Type: Object
  - **tooltip**Type: jQuery生成tooltip(工具提示框) 的元素。

Code examples:

初始化带有指定 close 回调的 tooltip(工具提示框)：

```
$( ".selector" ).tooltip({
  close: function( event, ui ) {}
});
```

绑定一个事件监听器到 tooltipclose 事件：

```
$( ".selector" ).on( "tooltipclose", function( event, ui ) {} );
```

create(event, ui)Type: `tooltipcreate`

在创建tooltip(工具提示框)时触发该事件。

- **event**Type: Event
- **ui**Type: Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

Code examples:

初始化带有指定 create 回调的 tooltip(工具提示框)：

```
$( ".selector" ).tooltip({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 tooltipcreate 事件：

```
$( ".selector" ).on( "tooltipcreate", function( event, ui ) {} );
```

open(event, ui)Type: `tooltipopen`

当tooltip(工具提示框)打开，显示时，触发此事件，触发的事件为 `focusin` 或 `mouseover` 。

- **event**Type: Event
- **ui**Type: Object
    - **tooltip**Type: jQuery生成tooltip(工具提示框) 的元素。

Code examples:

初始化带有指定 open 回调的 tooltip(工具提示框)：

```
$( ".selector" ).tooltip({
  open: function( event, ui ) {}
});
```

绑定一个事件监听器到 tooltipopen 事件：

```
$( ".selector" ).on( "tooltipopen", function( event, ui ) {} );
```

Example:

使用带有 **title** 属性的所有元素的事件代理，在文档上创建一个工具提示框（**Tooltip**）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>tooltip demo</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-ui.cs
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>
  <a href="#" title="Anchor description">Anchor text</a>
  <input title="Input help">
</p>
<script>
  $( document ).tooltip();
</script>

</body>
</html>
```